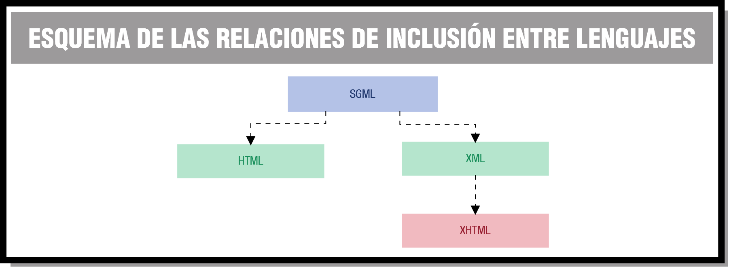
**DI02.- Elaboración de interfaces mediante documentos XML.**

**1.- Lenguajes basados en XML.**

**¿Qué es XML?**

**XML** (eXtensible Markup Languaje o Lenguaje de Marcado eXtensible) es un lenguaje basado en texto de definición de documentos, que permite representar información estructurada de gran variedad de documentos.

Al estar puramente basado en texto, un documento XML será un [archivo plano](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) en el que, a través de una serie de elementos propios se definen aspectos del documento a generar, pero no de su diseño propiamente dicho, sino de su estructura.

**Orígenes**

Hacer independiente la estructura de un documento de la constante evolución de las herramientas que se usan para visualizarlo ha sido, desde hace tiempo, foco de interés de los desarrolladores y desarrolladoras de aplicaciones informáticas. En los años sesenta, Charles F. Goldfab, por encargo de IBM desarrolló el lenguaje GML, Generalized Markup Languaje, cuyo objetivo era describir la estructura de los documentos de forma que el resultado no dependiese de una determinada plataforma o una aplicación específica. De la evolución de GML surgió SGML que se convirtió en el estándar internacional ISO 8879. Sin embargo tuvo difusión tan sólo en medios académicos y de la administración, pero no tuvo demasiado éxito entre los usuarios medios debido a su dificultad.

El hito que supuso la amplia difusión de las tecnologías de la comunicación y la circulación de documentos electrónicos entre cualquier punto del mundo fue la creación de la World Wide Web (Tim Berners-Lee, en 1990), y, asociado a ello la aparición de [HTML](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/), lenguaje de descripción de documentos basado en SGML para la web.

HTML es un lenguaje que permite combinar en un solo documento el contenido con el formato, por ejemplo:

* Si en un documento HTML se añade el siguiente código: "El padre del lenguaje **GML** fue **Charles F. Golfab**, que lo desarrolló por encargo de **IBM**."
* Conseguiré algo parecido a esto: El padre del lenguaje **GML** fue **Charles F. Golfab**, que lo desarrolló por encargo de **IBM**.

Ya que encerrar un texto entre **y** equivale a aplicar el formato negrita. Estos elementos de HTML como **y se denominan** [**etiquetas**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/)**, cada etiqueta tiene un significado en el diseño del documento final o en su estructura.**

**XML, sin embargo, no incluye ninguna orientación al diseño es puramente un lenguaje estructural cuyo objetivo es definir cómo se organiza la información en un documento, el diseño se definirá después, de esta forma se hace independiente al documento de la plataforma, e incluso de la aplicación con la que se va a visualizar (Web, impreso, como documento de texto, y más). Por ejemplo, si definimos una página web con XML podremos verla fácilmente en un navegador web, en un teléfono móvil, o, incluso, en un lector Braille, sin modificar el documento de base. Lo que cambia es la forma en que se interpreta el contenido.**

**Reflexiona**

**XML no incluye ninguna orientación al diseño. Esto quiere decir que no vamos a hacer referencia a ningún aspecto relativo a la aplicación final en la que visualicemos el documento. Entonces ¿cómo crees que podemos usar el mismo archivo para ver en un navegador que en un teléfono móvil? ¿Qué necesitaremos para visualizar ese contenido de formas diferentes?**

**Retroalimentación:**

**Efectivamente, los archivos XML se pueden visualizar en diferentes plataformas, como navegadores web o teléfonos móviles, para salvar las diferencias se debe acompañar de una archivo auxiliar, que interpreta cómo debe visualizarse el contenido XML en cada dispositivo. Este archivo se denomina XSL y contiene la definición de cómo se verán los elementos XML en un dispositivo concret**

**Autoevaluación**

**Pregunta**

¿Cual de las siguientes afirmaciones es correcta?.

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-6_63): SGML ha sido un lenguaje de muy amplia difusión desde su creación en los años ochenta.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-6_74): La estructura de un documento XML depende totalmente de la aplicación con la que se visualizará.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-6_77): El lenguaje HTML es un lenguaje XML puro.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-6_80): Con HTML podemos indicar cuestiones tales como el color de la fuente o si vamos a visualizar un texto centrado.

**Respuesta**: Opcion4. Es correcta ya que, a pesar de estar basado en XML, HTML si hace referencia a cuestiones relativas al diseño de la página web como por ejemplo el formato de la fuente o el párrafo

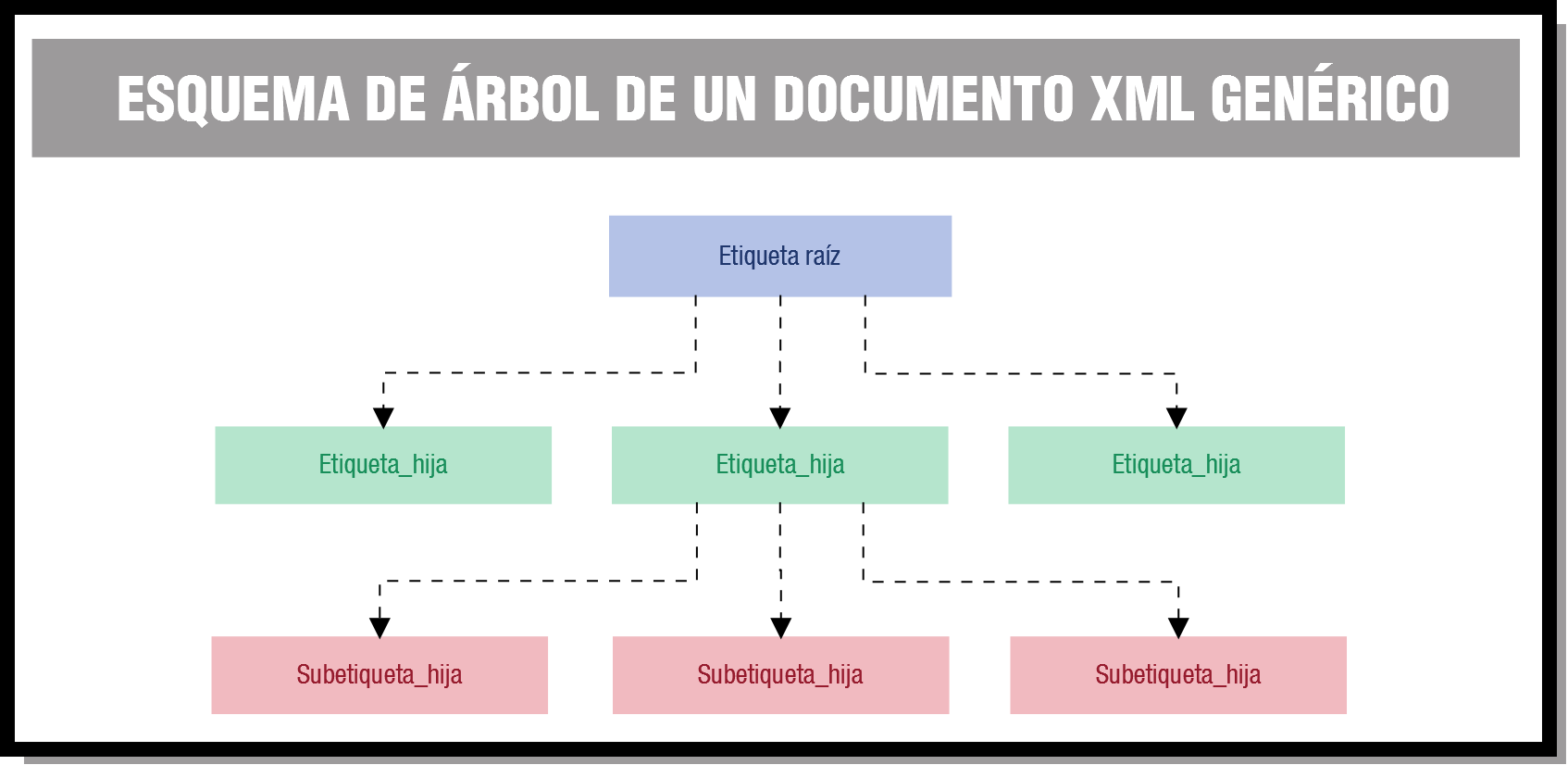
**Para saber más**

**Puedes encontrar un completo artículo sobre XML en este enlace:**

[**Artículo sobre XML.**](http://recursostic.educacion.es/observatorio/web/es/software/programacion/675-xml)

**1.1.- Etiquetas.**

**XML es un tanto parecido a HTML en el sentido de que utiliza etiquetas para definir elementos dentro de una estructura, con la diferencia de que cuando usamos HTML se trabaja con una serie de etiquetas predefinidas, y en XML se pueden definir según convenga a las necesidades del proyecto.**

****

**Las etiquetas están delimitadas por ángulos (<,>) e identifican el contenido que delimitan. Pueden tener atributos. Siguen la estructura:**

**Contenido**

**Las etiquetas en un documento XML tienen siempre etiqueta de cierre, de manera que toda la información referente al elemento queda comprendida entre ambas.**

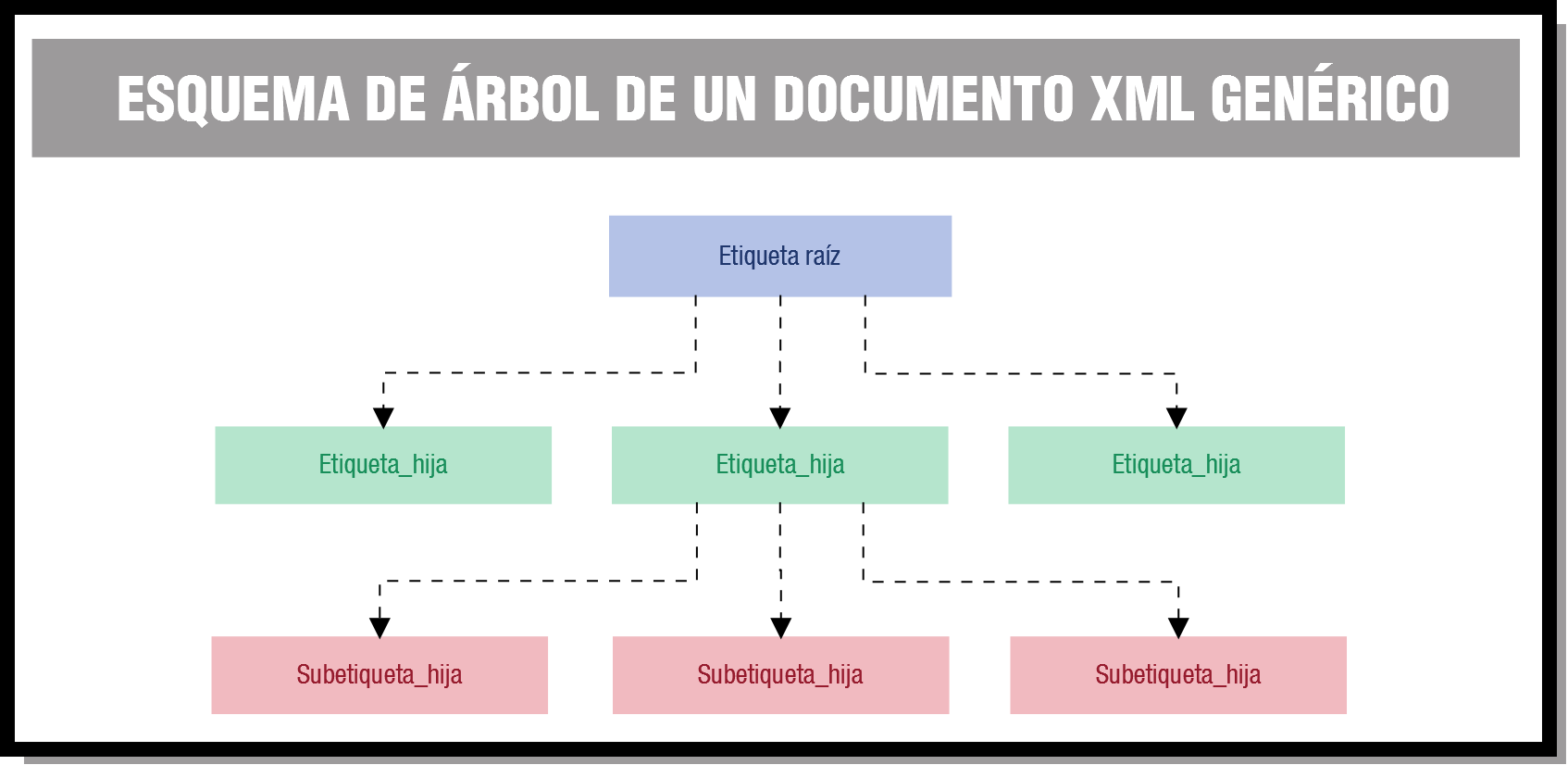
**Siempre existe una etiqueta raíz, que hace referencia al tipo de objeto que se describe. El resto son características o elementos del objeto, y se colocan anidadas dentro de la etiqueta raíz como etiquetas hijas. Por eso decimos que un documento XML tiene estructura de árbol.**

**Las etiquetas en un documento XML tienen siempre etiqueta de cierre, de manera que toda la información referente al elemento queda comprendida entre ambas.**

**Siempre existe una etiqueta raíz, que hace referencia al tipo de objeto que se describe. El resto son características o elementos del objeto, y se colocan anidadas dentro de la etiqueta raíz como etiquetas hijas. Por eso decimos que un documento XML tiene estructura de árbol.**

| **Estructura de árbol de un documento XML genérico** | |
| --- | --- |
| **Documento XML** | |
| <Etiqueta\_raiz>  <etiqueta\_hija>  <subetiqueta\_hija>...</subetiqueta\_hija>  <subetiqueta\_hija>...</subetiqueta\_hija>  ...  </etiqueta\_hija>  </Etiqueta\_raiz> |  |

**Estructura de árbol**

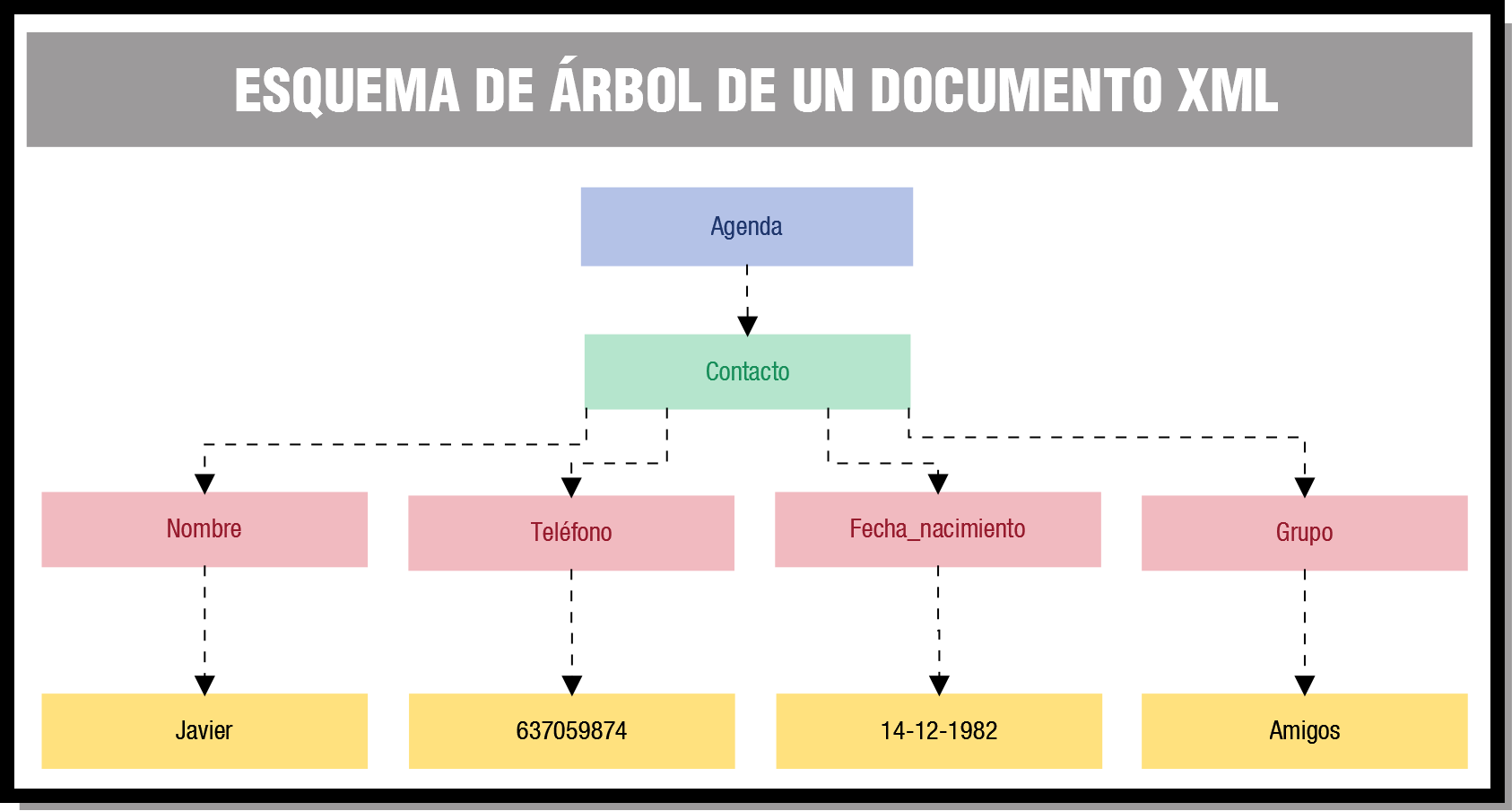
****

**Por ejemplo, si queremos usar XML para almacenar la información de nuestra agenda de contactos podríamos definir una etiqueta llamada contacto para cada entrada de la agenda, y dentro de los contactos otras etiquetas para el nombre, teléfono, fecha de nacimiento, grupo, entre otras. Si tengo un amigo que se llama Javier, cuyo número de teléfono es 637-059874 y que nació el 14 de diciembre de 1982, almacenaría la información referente a él en mi agenda de la siguiente manera:**

**dato, el contacto tiene cuatro características, su nombre es Javier, el teléfono 637059874, la fecha de nacimiento es 14-12-1982 y el grupo Amigos. En la segunda columna aparece como encabezado, en la primera fila, el título "Estructura de árbol" y, en la segunda fila, la imagen del árbol del documento, en el que aparece el nodo raíz que tiene como texto Agenda, del que parte un nodo con el texto Contacto, del que, a su vez, salen cuatro nodos con los nombres de las características, Nombre, Teléfono, Fecha\_nacimiento y Grupo, y de cada uno de ellos otro nodo con el dato correspondiente, respectivamente, Javier, 637059874, 14-12-1982 y Amigos.">**

| **Estructura de árbol del documento XML para la agenda** | |
| --- | --- |
| **Documento XML** | |
| <agenda>  <contacto>  <nombre>Javier</nombre  <teléfono>637059874</teléfono>  <fecha\_nacimiento>14-12-1982</fecha\_nacimiento>  <grupo>Amigos</grupo>  </contacto>  </agenda> |  |

**Estructura de árbol**

****

**Donde la etiqueta raíz sería agenda, contacto sería una etiqueta hija y nombre, teléfono, fecha\_nacimiento y grupo funcionarían como subetiquetas.**

**1.2.- Atributos y valores.**

**Se dice que un elemento XML tiene contenido cuando se ha añadido algún texto entre la etiqueta de apertura y cierre. Sin embargo pueden darse casos en los que no se pueda almacenar toda la información pertinente del contenido sólo en el texto que encierran las etiquetas.**

**Cuando necesitamos añadir información adicional a un elemento XML de alguna manera modificamos la etiqueta añadiéndole atributos.**

**Contenido**

**Los atributos permiten proporcionar información adicional sobre el elemento. Por ejemplo, puedo precisar si el teléfono que he guardado para mi amigo Javier es su teléfono fijo, su móvil o el teléfono del trabajo añadiendo el atributo tipo:**

**<agenda>**

**<contacto>**

**<nombre>Javier</nombre**

**<teléfono tipo=”móvil”>637059874</teléfono>**

**<fecha\_nacimiento>14-12-1982</fecha\_nacimiento>**

**<grupo>Amigos</grupo>**

**</contacto>**

**</agenda>**

**No siempre ocurre que cumpliendo los requisitos anteriormente comentados el documento XML sea correcto. Para asegurarnos de que un documento XML esté bien formado debe cumplir las siguientes reglas:**

* **Debe existir un elemento raíz.**
* **Todos los elementos XML deben tener su correspondiente etiqueta de cierre.**
* **Es sensible a mayúsculas.**
* **El anidamiento debe hacerse conforme a la estructura del árbol del documento. Es decir, si abro la etiqueta y a continuación se cierran en sentido inverso, no puedo cerrar antes y después**
* **Los valores de los atributos van entrecomillados siempre.**

**A la hora de poner nombre a las etiquetas se deben seguir las siguientes recomendaciones:**

* **Se pueden usar letras, número y otros caracteres.**
* **No se puede empezar por un número o signo de puntuación.**
* **No se puede empezar por las letras XML.**
* **Los nombres no pueden contener espacios en blanco.**

**Autoevaluación**

Pregunta

Texto de la pregunta: ¿cuál es la expresión correcta?

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-14_63) : <fecha de nacimiento>14-12-1982</fecha de nacimiento>

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-14_92): <teléfono tipo=movil>637059874</teléfono>

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-14_95): <contacto> <nombre> Javier </contacto> </nombre>

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-14_98): <email tipo="personal">javier@gmail.com</email>.

**Respuesta**: Opcion4

**2.- Lenguajes de descripción de interfaces basados en XML.**

**El desarrollo de interfaces de usuario es una actividad compleja que requiere de la intervención de múltiples factores. El desarrollador o desarrolladora se enfrenta a una serie de hitos que debe superar:**

* **En primer lugar se debe definir lo que se espera de la interfaz, que requisitos debe cubrir y el entorno en el que se integra.**
* **También se debe tener en cuenta la variedad de herramientas y lenguajes de programación que exigen un elevado nivel de conocimientos.**
* **Por otra parte necesitamos interfaces (así como las aplicaciones subyacentes) para múltiples contextos de uso, que dependen del usuario (por su idioma, preferencias o posibles discapacidades), de la plataforma (ordenadores personales, móviles, miniportátiles, pantallas, etc.), del modo de presentación de la interfaz (gráfica o texto) o del dispositivo final (pantallas de ordenador, pizarras digitales, proyectores, etc.).**

**2.1.- Proceso de elaboración de las interfaces.**

**La primera posibilidad para desarrollar una interfaz suele ser emplear el mismo lenguaje, normalmente de alto nivel, que se usa para implementar la funcionalidad de la aplicación como Java, C#, etc. Tiene como ventaja la sencillez, puesto que no es preciso adquirir nuevos conocimientos, la creación de la interfaz se integra en el proceso de desarrollo de la aplicación y, normalmente, no es necesario trabajar con herramientas diferentes, ni en la programación ni posteriormente a la hora de** [**compilar**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **o ejecutar el programa. Sin embargo, tiene como principal desventaja la dependencia de la plataforma, del dispositivo y del propio lenguaje. Básicamente, la portabilidad de las interfaces creadas de esta manera serán las que proporcione el lenguaje.**

**La creación de interfaces de usuario usando** [**lenguajes de descripción**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **basados en XML pretende solventar esto, proporcionando un medio que permita construir interfaces mediante descripciones de alto nivel en los distintos aspectos de la interfaz: estructura y comportamiento, de modo que a partir de la descripción se pueda generar de forma automatizada la interfaz de usuario final. Además este proceso permite centrar el desarrollo en las necesidades del usuario, puesto que no se realiza siguiendo las directrices marcadas por el lenguaje.**

**En este ámbito se utilizará una notación de alto nivel para desarrollar la interfaz, que se almacena en un archivo aparte, en formato XML. Posteriormente se trata este archivo para obtener el código de la interfaz que se integrará en la aplicación. A este procedimiento se le denomina mapear la interfaz. El proceso de mapeado depende del lenguaje concreto que se esté usando.**

**Tras el correspondiente proceso se podrá visualizar la interfaz en un dispositivo final.**

**Reflexiona**

**El uso de tecnologías XML permite separar completamente la creación de la interfaz de la programación de la aplicación subyacente. ¿Qué ventajas crees que aporta esto? ¿Puede mejorar el procedimiento de desarrollo de aplicaciones con interfaces gráficas complejas?**

**Retroalimentación:**

**De hecho así es, utilizar tecnologías XML para la creación de interfaces tiene como principal ventaja que, después de planificar adecuadamente qué esperamos de la aplicación y qué información debe proporcionarle la interfaz, el equipo puede dividirse totalmente el trabajo de desarrollo, ya que ambas tareas pueden realizarse completamente por separado con el consiguiente ahorro de tiempo. Además las modificaciones que se pueden realizar influyen mucho menos en la aplicación global y suelen afectar tan solo al ámbito que corresponde. Por otra parte, al utilizar lenguajes y herramientas específicas, normalmente, se consiguen interfaces más elaboradas ya que se pone a nuestra disposición elementos concretos de este ámbito de la programación.**

**A continuación veremos algunos ejemplos de lenguajes para el desarrollo de interfaces basados en XML y como realizan el proceso de mapeado.**

**Autoevaluación**

Pregunta

¿Qué frase es correcta con respecto al desarrollo de interfaces?

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-20_63): Es un proceso sencillo porque la misma interfaz es válida para todos los usuarios.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-20_105): Una vez creada un interfaz en un lenguaje de alto nivel podremos reutilizarla para cualquier tipo de dispositivo sin necesidad de modificarla.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-20_108): Una interfaz descrita con un lenguaje basado en XML se integra directamente en la aplicación.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-20_111): El uso de interfaces descritas con lenguajes basados en XML permite desarrollar por separado la interfaz gráfica de una aplicación y comportamiento.

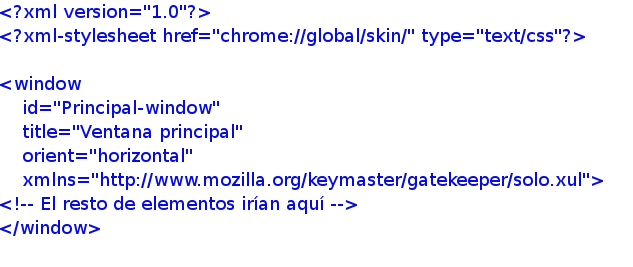
**Respuesta**: Opcion4. **Correcto, se puede dividir el trabajo en dos grupos, uno que genere los documentos con todas las interfaces y otro que desarrolle toda la funcionalidad, dejando la integración para el final del proyecto.**

**2.2.- XUL.**

**Es un lenguaje de marcas basado en XML para definir interfaces de usuario complejas, en las que podremos insertar los elementos habituales en un formulario, pero que también dispone de otros más complejos, como menús, barras de herramientas, estructuras jerárquicas o teclas de acceso directo. Así mismo, permite colocar los contenidos de la interfaz usando distribuciones complejas.**

**Este lenguaje se utiliza específicamente para desarrollar aplicaciones en red. Los documentos escritos en este lenguaje podrán visualizarse en los navegadores Mozilla y Netscape, de hecho la interfaz de ambos está desarrollada en XUL. Ambos navegadores funcionan sobre el motor de** [**renderizado**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **Gecko, que es la base que necesitamos para visualizar correctamente el contenido de un documento XUL. Gracias a esto, las aplicaciones XUL son mutiplataforma y multidispositivo, ya que siempre podremos encontrar un navegador Gecko para la plataforma y dispositivo deseados.**

**Uno de los principales atractivos de este lenguaje es que ofrece la posibilidad de crear complementos para el navegador Mozilla.**

****

**Por contra su principal desventaja es que no se podrá ejecutar en otros navegadores que, como Internet Explorer, no cumplan con esta característica.**

**Proceso de mapeado: Un documento XUL es interpretado para visualizar su contenido, en la renderización intervienen la definición de los elementos y las hojas de estilo. Es muy parecido a HTML, utiliza hojas de estilo para diseñar el aspecto y javascript para implementar el comportamiento de sus elementos. Para definir una interfaz se especifican tres grupos de componentes distintos:**

* **Content: Aquí se encuentran los documentos XUL, que definen el diseño de la interfaz. Incluye las etiquetas y la referencia a los documentos javascript.**
* **Skin: Contiene las hojas de estilos** [**(CSS)**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **y las imágenes, las cuales definen la apariencia de la interfaz.**
* **Locale: Los documentos** [**DTD**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **se encuentran aquí, estos documentos facilitan la localización de páginas XUL.**

**Cada uno de estos componentes se almacena en un archivo diferente, que puede ser editado.**

**Este es un ejemplo de archivo XUL, puedes encontrar un ejemplo un poco más extendido en este enlace:**

[**Ejemplo de XUL**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R09_ejemplo_XUL.txt)**.**

**Para saber más**

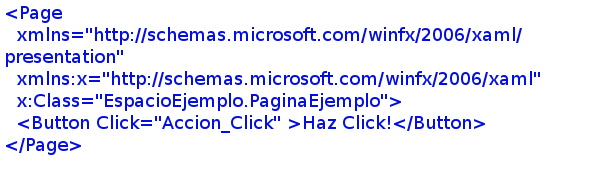
**Página oficial de XUL donde poder encontrar la referencia del lenguaje, ejemplos de aplicaciones. Cabe destacar el enlace a la aplicación XULRunner que te permitirá crear y ejecutar aplicaciones XUL.**

[**Página oficial de XUL.**](https://developer.mozilla.org/es/XUL)

**En este enlace encontrarás un pequeño tutorial, muy sencillo de seguir, para ver como funciona una aplicación web muy básica escrita en XUL.**

[**Mi primera aplicación XUL.**](http://es.wikibooks.org/wiki/XUL)

**2.3.- XAML.**

****

**Es un lenguaje de marcas empleado para la creación de interfaces en el modelo de programación .NET** [**Framework**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **de Microsoft. Las aplicaciones creadas podrán ejecutarse en entornos Windows.**

**Mediante XAML se pueden crear lo que se conoce como** [**RIA**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **(Rich Interface Applications) que contienen abundante material gráfico. XAML consta de una serie de elementos XML para representar los principales componentes gráficos, así como la distribución, paneles y manejadores de eventos. Puede hacerse programando directamente la interfaz mediante un editor de texto, o mediante el entorno de desarrollo gráfico incluido en la herramienta Expresión Blend. El código asociado a la interfaz es puramente declarativo, es decir, hace referencia tan solo al aspecto visual, pero no añade funcionalidad, salvo ciertas respuestas muy sencillas a interacciones con el usuario.**

**La realización de cálculos se añaden en un archivo independiente. Esto permite al equipo de desarrollo y al de diseño trabajar por separado, sin interferir mutuamente en su trabajo.**

**Proceso de mapeado: se realiza en tiempo de ejecución, los elementos de la interfaz se conectan con objetos del framework .NET y los atributos con propiedades de estos objetos para integrarlos en la aplicación. Para facilitar la traducción de XAML a código .NET se ha creado una relación para cada elemento XAML con una clase de la plataforma .NET.**

**Tienes el código del ejemplo en el siguiente documento:**

[**Documento con ejemplo para XAML.**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R12_ejemplo_XAML.txt)

**Para saber más**

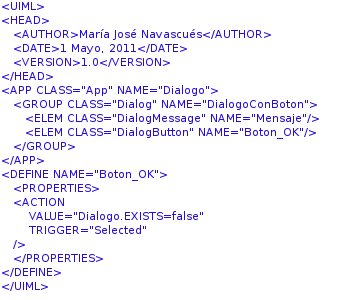
**Si quieres conocer algo más sobre XAML te sugerimos una página donde poder encontrar una introducción a XAML.**

[**Página oficial de XAML.**](http://msdn.microsoft.com/es-es/library/ms752059.aspx)

**Y también puedes visitar la siguiente página donde poder encontrar la referencia del lenguaje y ejemplos de aplicaciones dentro del ámbito del entorno de desarrollo Expression Blend.**

[**Página de la MSDN de XAML.**](http://msdn.microsoft.com/es-es/library/cc295215.aspx)

**2.4.- UIML.**

**Este lenguaje permite generar interfaces que son independientes de la plataforma y el lenguaje subyacente. Para generar una interfaz UIML se precisa crear un documento XML con la definición de la interfaz, utilizando los elementos de UIML y una hoja de estilos que traslade esa definición a la plataforma y lenguaje seleccionado, de esta forma para una aplicación concreta tan solo necesitaremos un documento UIML, y tantas hojas de estilo para la traducción como nos sean necesarias.**

**Este sistema tiene como principal ventaja que solo se precisa un único diseño de la interfaz de la aplicación, independientemente del dispositivo donde será visualizado, con lo que evitamos el peligro de desarrollar interfaces para dispositivos que no estén en el mercado en el futuro.**

**UIML describe una interfaz en tres niveles: presentación, contenido y lógica.**

**La presentación se refiere a la apariencia de la interfaz; el contenido se refiere a los componentes de la interfaz y la lógica se refiere a la interacción usuario – interfaz (eventos del ratón, teclado...).**

**En UIML, una interfaz de usuario es una jerarquía de elementos XML. Cada uno de los componentes de una interfaz (botones, cajas de texto, etiquetas...) son una entidad XML. Estas entidades son elementos Part. Cada uno tiene asociado un elemento Content, que puede ser texto, imagen, etc.**

**Para definir el comportamiento de la interfaz se realiza el mapeo de las partes a los elementos correspondientes del lenguaje de implementación elegido y la conexión con la lógica de aplicación.**

**Estos bloques facilitan la separación entre los elementos que componen la interfaz, distinguiendo entre el modelado estructural, la visualización y el modelado del comportamiento.**

**El lenguaje UIML permite la traducción automática al lenguaje utilizado por el dispositivo final. El proceso de traducción se realiza en el propio dispositivo o en el servidor de la interfaz dependiendo del dispositivo del que se trate.**

**Tienes el código del ejemplo en el siguiente enlace:**

[**Ejemplo de UIML.**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R14_ejemplo_UIML.txt)

**Para saber más**

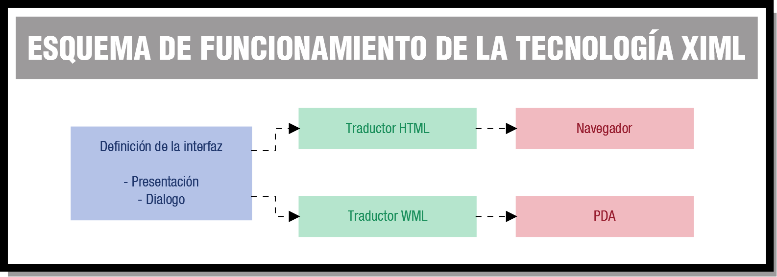
**Aquí tienes un enlace donde poder encontrar un tutorial del lenguaje.**

[**User Interface Markup Language (UIML).**](http://xml.coverpages.org/uiml20-17Jan00.pdf)

**En este enlace encontrarás una descripción del lenguaje y ejemplos.**

[**UIML: An Appliance-Independent XML User Interface Language.**](http://www.ra.ethz.ch/CDstore/www8/data/2170/pdf/pd1.pdf)

**2.5.- XIML.**

****[**XIML**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **o Lenguaje Extensible de Formato para Interfaces, es un lenguaje de desarrollo de interfaces cuyo objetivo es cubrir todo el ciclo de vida del software, incluyendo las fases de diseño, operación y evaluación, por lo tanto, además de proporcionar la infraestructura para poder diseñar una interfaz gráfica de usuario con cierto nivel de complejidad, también proporciona herramientas para atender a todo el proceso de desarrollo de la misma.**

**XIML trabaja con los elementos abstractos y concretos de una interfaz de usuario. Los elementos abstractos hacen referencia al contexto en el que se interacciona con la interfaz, y se representan en:**

* **Tareas: procesos o tareas de usuario que puede ejecutar la interfaz.**
* **Dominio: conjunto de objetos y clases con los que se opera desde la interfaz. Están distribuidos jerárquicamente.**
* **Usuarios: también se organizan jerárquicamente y representan usuarios o grupos de usuarios que hacen uso de los datos y procesos en las tareas.**

**Los elementos concretos hacen referencia a aquello que se representa físicamente en la interfaz, como los controles de formulario, y se representan en:**

* **Presentación: colección jerárquica de elementos que interaccionan con el usuario desde la interfaz, puede ser desde una ventana a un botón, pasando por controles más complejos como un control ActiveX.**
* **Dialogo: colección jerárquica de acciones de interacción que permiten al usuario comunicarse con los elementos de la interfaz.**

**Los elementos de la interfaz interaccionan a través de relaciones en las que se indica que elementos intervienen y el tipo de relación que hay entre ambos.**

**Proceso de mapeado: En este lenguaje se separa completamente la definición de la interfaz de la forma en que es renderizada. Precisa de un traductor que traslade la definición a código visible para cada dispositivo específico en el que se quiera usar la interfaz.**

**De esta forma se consigue un lenguaje completamente multiplataforma.**

**Autoevaluación**

Pregunta

Texto de la pregunta: Los lenguajes de desarrollo de interfaces basados en XML...

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-31_63): ...son lenguajes de programación de alto nivel.

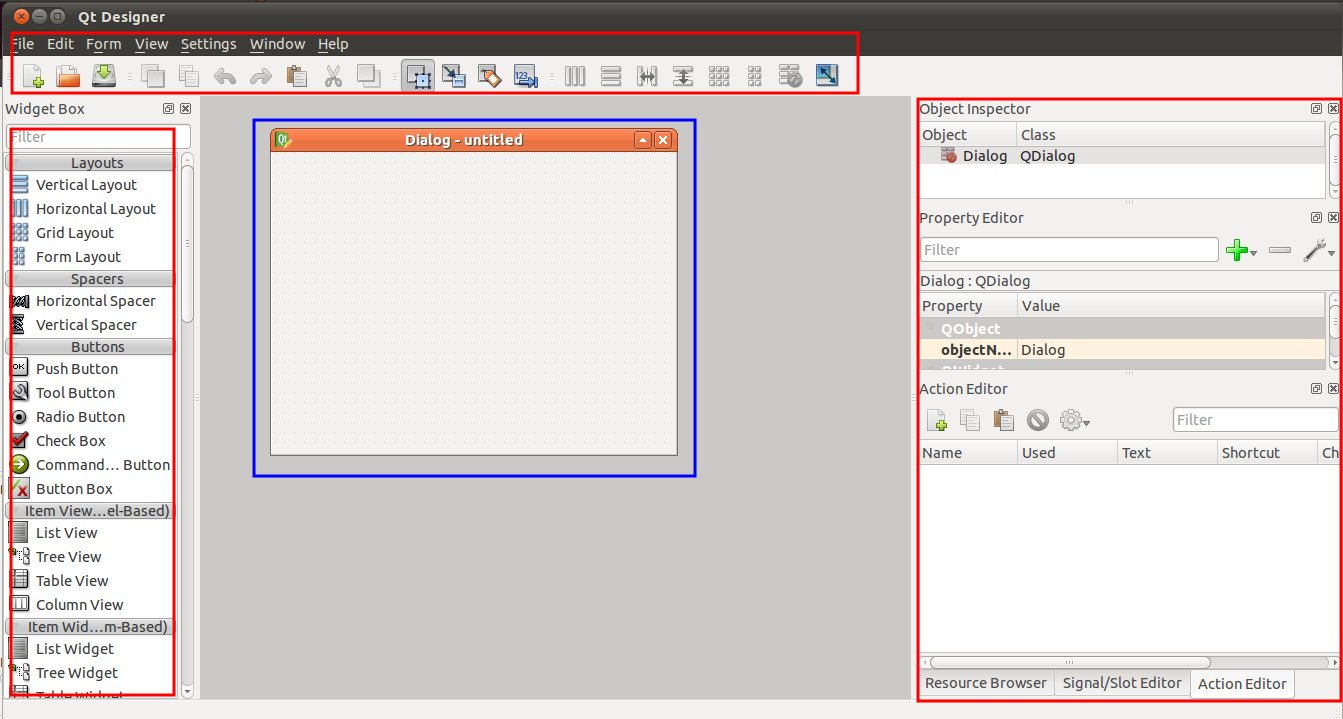
[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-31_125): ...permiten desarrollar aplicaciones completas siempre sin necesidad de que intervengan otros lenguajes de programación.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-31_128): …solo permiten generar aplicaciones para Internet.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-31_131): ...necesitan que, posteriormente, se mapeen los elementos XML a objetos que entienda el dispositivo final.

**Respuesta**: Opcion4. **Es correcto, normalmente el archivo XML sufre un proceso de análisis antes de ser renderizados o integrados en una aplicación.**

**3.- Herramientas para la creación de interfaces multiplataforma.**

**Existen muchos tipos de software para la creación de interfaces de usuario. Habitualmente estas herramientas tienen en común que para generar interfaces gráficas usan un sistema de ventanas, las cuales permiten la división de la pantalla en diferentes regiones rectangulares, llamadas "ventanas" cuya parte central es el conjunto de herramientas (toolkit).**

**Cuando creamos una aplicación con interfaces gráficas, en primer lugar hay que diseñar la interfaz, normalmente el** [**toolkit**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **contiene los objetos gráficos más empleados, tales como menús, botones, etiquetas, barras de scroll, y campos para entrada de texto que compondrán la interfaz, mientras que el sistema de ventanas provee de procedimientos que permiten dibujar figuras en la pantalla y sirve como medio de entrada de las acciones del usuario. En la imagen puedes apreciar los conjuntos de herramientas enmarcados en rojo (en la imagen cuadro derecho y barra de herramientas superior) y la zona de dibujo en azul (en la imagen cuadro central).**

**A continuación se añade funcionalidad a los elementos de la interfaz a través de una serie de procedimientos definidos por el programador o programadora. La función de estos procedimientos es el decidir la forma en que se comportarán los objetos gráficos.**

**Por último, se conecta la interfaz generada con la aplicación de destino. Esto se puede realizar de diferentes maneras. Si usamos un único lenguaje de programación para la interfaz y la funcionalidad, lo usual es contar con un entorno de desarrollo integrado común para todas las fases de desarrollo como al utilizar la biblioteca swing con Java. Sin embargo, cuando el lenguaje final es uno y el lenguaje de la interfaz otro, como es el caso de las tecnologías basadas en XML que estamos viendo, normalmente se requiere de un proceso de traducción de los elementos XML a elementos que entienda la plataforma final, como en el caso de XAML que traduce cada elemento XML a clases de la plataforma .NET. En el caso de XUL es el motor de renderización el que se encarga de hacer las transformaciones necesarios para visualizar la interfaz. Este proceso puede ser más o menos automático en función de las tecnologías seleccionadas.**

**A continuación veremos algunos ejemplos de herramientas existentes, tanto libres como propietarias, que difieren en la manera que tratan los archivos XML con las interfaces para incluirlos en la aplicación final.**

**Autoevaluación**

Pregunta

Texto de la pregunta: Las herramientas para la generación de interfaces...

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-34_63): ...son muy complicadas de aprender.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-34_137): ...no se integran nunca con otras herramientas de desarrollo de aplicaciones.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-34_140): ...se usan para editar la interfaz y poder modificarla en modo texto.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-34_143): ...permiten al programador o programadora concentrarse en los aspectos relativos al diseño.

**Respuesta**: Opcion4. **Correcta, al abstraernos de la sintaxis y aspectos internos de cada lenguaje de modelado conseguimos centrarnos en el diseño de la interfaz.**

**3.1.- Presentación de algunas herramientas.**

****

**El principal objetivo de estas herramientas es ocultar la sintaxis de los lenguajes de modelado y proporcionarles una interfaz que permita especificar adecuadamente el modelo de interfaz en los tres aspectos que hemos visto, a saber:**

**La interfaz se almacena en un archivo de texto plano siguiendo las directrices del estándar XML.**

**Estas herramienta disponen de editores, intérpretes, generadores y otras aplicaciones útiles para llevar a cabo tareas relacionadas con la elaboración, manipulación o generación de modelos de interfaz.**

**Entre otras, con estas características podemos encontrar las siguientes aplicaciones:**

* **Libres:** 
  + **QT Designer.**
  + **Glade.**
* **Propietarias:** 
  + **Expression Blend de Microsoft.**
  + **Flex de Adobe.**

**Debes conocer**

**Aquí tienes algunos vídeos con las características más importantes de las herramientas mencionadas en el texto.**

[**Mostrar**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/blocks/recopila/recopila.php?id=3415&dopt=1)

[**Herramientas para la generación de interfaces gráficas de usuario basadas en XML**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/blocks/recopila/recopila.php?id=3415&dopt=1)

[**Herramientas libres**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/blocks/recopila/recopila.php?id=3415&dopt=1)

**QT Designer**

**GLASE**

[**Herramientas propietarias**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/blocks/recopila/recopila.php?id=3415&dopt=1)

**FLEX de Adobe**

**Microsoft Expression Blend**

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Presentacion_Herramientas_para_desarrollo_de_interfaces_basadas_XML.html)

**Para saber más**

**Si tienes curiosidad por saber algo más de estas herramientas aquí tienes lo enlaces que te llevarán a sus páginas oficiales:**

[**Página oficial de QT.**](https://qt-project.org/)

[**Página oficial de GLADE.**](http://glade.gnome.org/)

[**Página oficial de FLEX.**](http://www.adobe.com/es/products/flex/)

[**Página oficial de Microsoft Expression Blend.**](http://www.microsoft.com/spain/expression/products/Purchase.aspx?key=blend)

[**Página de MSDN para Expressión Blend.**](http://msdn.microsoft.com/es-es/library/ee341599.aspx)

**Autoevaluación**

Pregunta

Texto de la pregunta: ¿Qué es una aplicación RIA?

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-39_63): Aquella que solo ejecuta en un terminal.

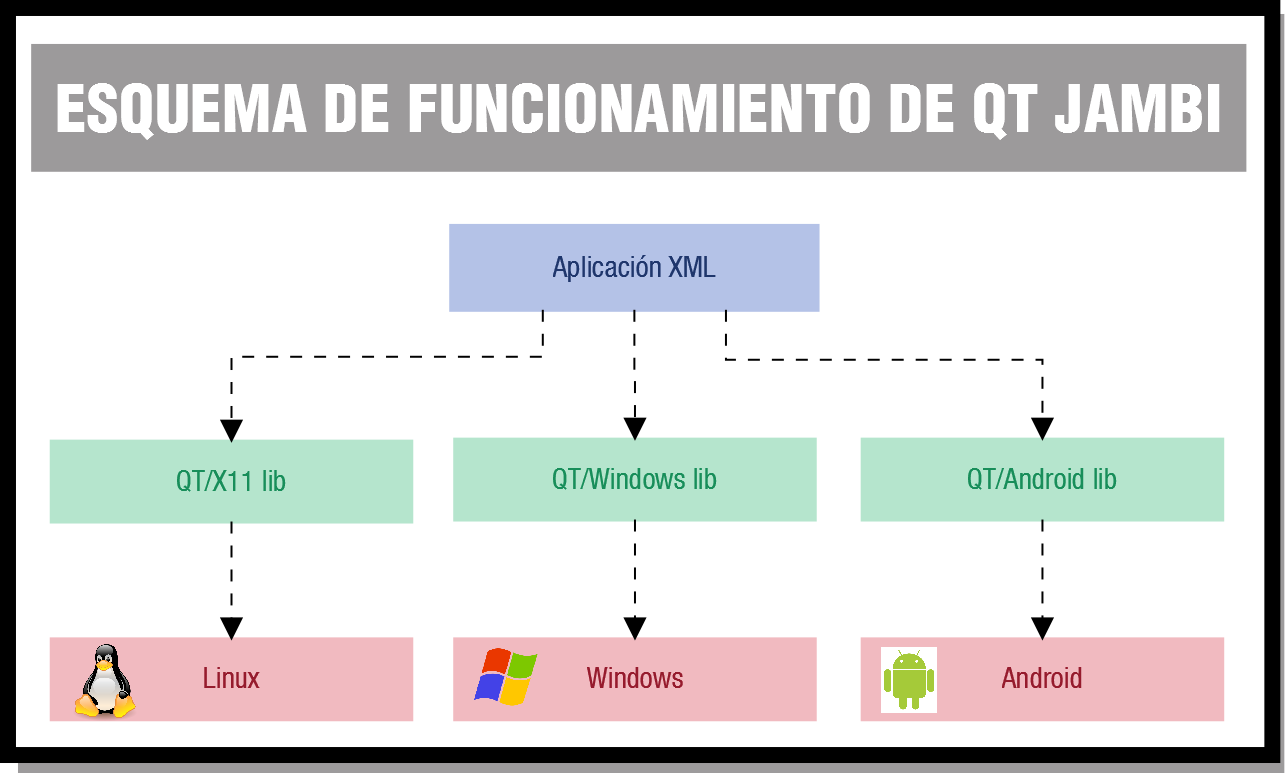
[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-39_151): Un programa con una interfaz de usuario compleja.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-39_154): Una aplicación para realizar operaciones bancarias.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-39_157): Aquella que se ejecuta en un navegador web y su aspecto y funcionalidad es el de una aplicación de escritorio.

Respuesta: Opcion4. **Correcta, por eso es tan importante la creación de interfaces multiplataforma.**

**4.- Ejemplo de desarrollo de una interfaz básica con QT Designer.**

**QT es un entorno de desarrollo de aplicaciones multiplataforma para la creación de aplicaciones web, de escritorio y para móviles. Consta de un conjunto de herramientas de desarrollo y un API, las bibliotecas QT, escrito en C++.**

**El** [**IDE**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **se denomina QT Creator y dispone de un conjunto de módulos para el desarrollo completo de aplicaciones de diversos tipos, entre ellas nos centraremos en QT Designer que es una herramienta de tipo WYSIWYG para la elaboración de interfaces de usuario complejas basadas en componentes de las bibliotecas QT. Se pueden incluir todo tipo de elementos como menús, paneles, barras de herramientas, y disponerlos usando distribuciones muy completas.**

**Entre todas las herramientas disponibles para el desarrollo de interfaces basadas en tecnologías XML, se ha optado por QT Designer por varios motivos:**

* **Está disponible bajo licencia LPGL, aunque combina con una licencia comercial.**
* **Es una herramienta multiplataforma, estando soportada por las principales plataformas del mercado (Windows, Linux, MACOS) así como otras más emergentes, como Windows Mobile, Symbian, Maemo, o, más reciente, Android.**
* **Produce código portable.**
* **Utiliza código intermedio XML para definir las interfaces de las aplicaciones.**
* **Es posible integrarlo de forma sencilla con las herramientas que estamos usando para realizar las prácticas en este ciclo e involucrar el desarrollo de interfaces en XML dentro de proyectos de mayor envergadura sin necesidad de aprender otras nuevas.**

**Breve historia.**

**QT apareció como biblioteca en 1992 para Windows y X11, creada por Haavard Nord y Eirik Chambe-Eng. Más tarde, en 1994 surge la empresa Troll Tech (antes Quasar Technologies), encargada de su desarrollo. En esa época QT se involucra en el desarrollo del escritorio KDE. No todo el mundo estaba de acuerdo en que una plataforma que no era totalmente libre participara en el proyecto GNU, lo que hizo que paulatinamente fueran surgiendo licencias LGPL para todas las plataformas. En el año 2008 fue adquirida por Nokia, y comenzó su expansión por la plataformas móviles (Windows mobile, Symbian), liberando en febrero de 2011 la versión Alpha para Android, con lo que se convierte en uno de los entornos de desarrollo más completos con respecto a dispositivos finales cubiertos.**

**4.1.- El binding QT Jambi.**

**Las bibliotecas QT están desarrolladas en y para el lenguaje C++, sin embargo nosotros vamos a trabajar con Java. Para hacerlo posible utilizaremos un binding que es un traductor que permite a desarrolladores y desarrolladoras de determinados lenguajes utilizar las tecnologías QT, así como a los programadores y programadoras QT exportar su trabajo a otros lenguajes. El** [**binding**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/) **QT/Java, que se lama QT Jambi, permite a los desarrolladores Java utilizar las tecnologías QT mediante una capa intermedia de código estático que transmite las llamadas a la API de Java a la biblioteca compilada QT de C++**

**Reflexiona**

**Los bindings extienden el uso de bibliotecas escritas en un lenguaje determinado a otros lenguajes de programación. El código QT Jambi será Java por completo, sin que el programador tenga que preocuparse de la arquitectura de las clases QT en C++.**

**Debes conocer**

**Para el desarrollo del ejemplo necesitamos tener instaladas las bibliotecas y herramientas del binding QT-Jambi.**

[**QT Jambi.**](https://en.wikipedia.org/wiki/Qt_Jambi)

[**Descarga QT Jambi.**](https://sourceforge.net/projects/qtjambi/)

[**Tutoriales QT Jambi.**](https://www.google.com/search?q=qtjambi&client=ms-android-vf-es-revc&prmd=mvin&sxsrf=ACYBGNSAz97yHmdkAsal4aoUmVqJJpbosg:1581027099173&source=lnms&tbm=vid&sa=X&ved=2ahUKEwi3-oD1-L3nAhWQEVAKHZoVCbUQ_AUoAnoECA8QAg&biw=360&bih=664&dpr=3#ip=1)

**El desarrollo de las aplicaciones una vez generada la interfaz se hará utilizando NetBeans que se presupone ya instalado, así como una versión del JDK de Java superior a la versión 5.1.**

**En el Anexo I y II tienes un documento con el proceso de instalación de QT Jambi.**

**Para saber más**

**QT es un entorno de desarrollo de aplicaciones multiplataforma en lenguaje C++ muy completo. Se ha usado para desarrollar, entre otras cosas, el escritorio KDE de Ubuntu. Si quieres saber algo más de la herramienta QT Creator y el API en general sigue el enlace a la página principal de QT.**

[**Página principal de QT.**](https://www.qt.io/)

**Autoevaluación**

Pregunta

¿Qué es un binding?

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-47_63): Una aplicación para la elaboración de interfaces gráficas de usuario.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-47_164): Un módulo de las bibliotecas QT para conectar la interfaz gráfica con el código que programa su comportamiento.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-47_167): Una función C++.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-47_170): Un traductor que permite a programadores en java utilizar las bibliotecas QT.

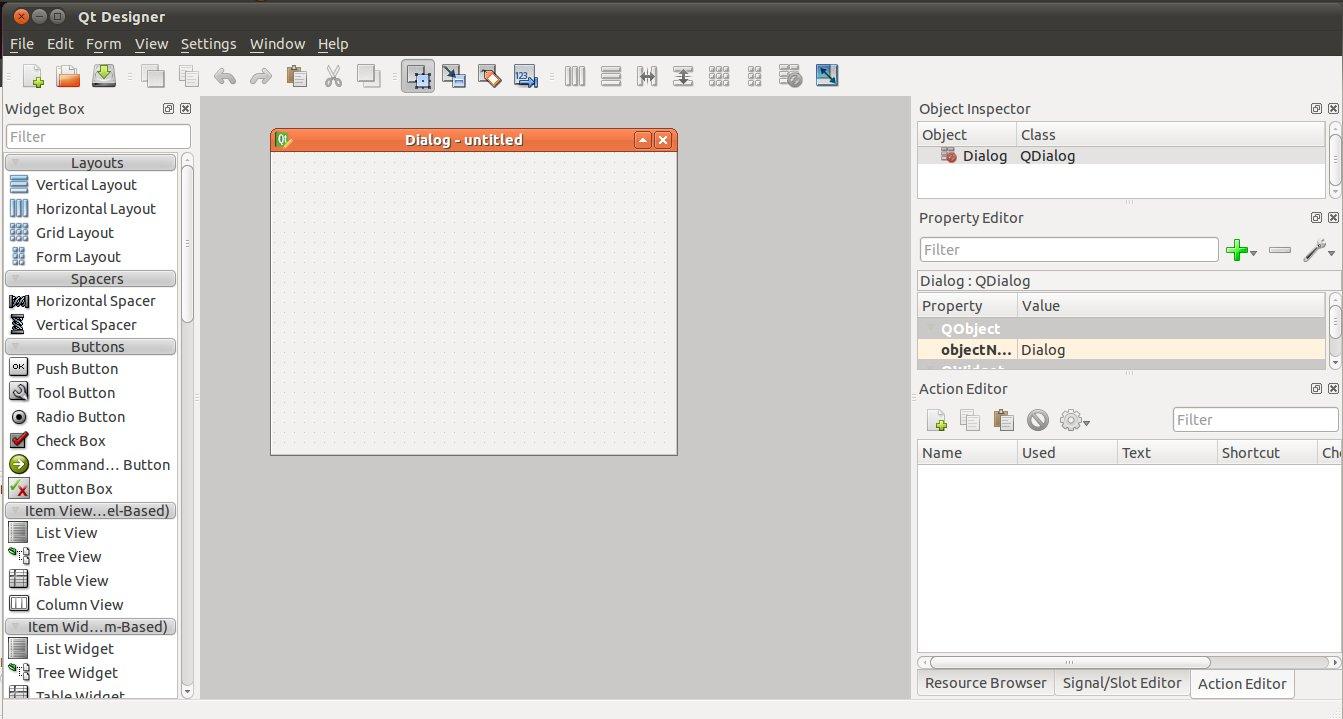
**Respuesta**: Opcion4. **Correcto. Es un código intermedio estático que conecta las llamadas a la API java con la biblioteca QT escrita y compilada en C++.**

**4.2.- Revisión general del entorno.**

**Una vez realizada la instalación de QT Jambi vamos revisar la herramienta para el desarrollo gráfico de interfaces de usuario QT Designer. Se ejecuta desde la línea de comandos llamando al archivo designer.sh que se encuentra localizado en el directorio donde hemos desempaquetado qtjambi:**

**usuario@equipo:~/qtjambi$ ./designer.sh**

**Su aspecto es el siguiente:**

****

**Cuenta en la zona superior con un menú de acceso a las diferentes opciones para gestionar archivos, portapapeles, ayuda, etc., así como una barra de herramientas justo debajo con atajos a las opciones más comunes:**

* **Relacionadas con la gestión de archivos: nuevo, abrir y guardar.**
* **Enviar al fondo o al frente.**
* **Relacionados con los modos de edición del programa: edición de** [**widgets**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/)**, edición de signal-slot, edición de buddies y edición del orden de tabulación, que veremos más adelante.**
* **Relacionados con los elementos de distribución de objetos en el formulario ([layout](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/)).**

**Se puede modificar la barra de herramientas desde el menú View >> Toolbars >> Configure Toolbars.**

**A la izquierda se encuentra el Widget Box, que ofrece una selección de widgets estándar Qt, layouts, y otros objetos que se pueden utilizar para crear interfaces de usuario. Los widgets se agrupan en categorías según su uso o características.**

**A la derecha encontramos un conjunto de paneles que permiten gestionar los widgets incluidos en la interfaz, los más importantes son:**

* **Object Inspector (Inspector de objetos): contiene la estructura jerárquica de widgets que se han añadido al formulario que se está creando.**
* **Property Editor (Editor de propiedades): permite la edición de todas las propiedades del widget seleccionado en el inspector de objetos.**
* **Editor de Signal/Slot: el mecanismo Signal-slot se emplea para asociar comportamientos a los elementos de la interfaz.**
* **Editor de acciones: se utiliza para la gestión de acciones. Las acciones se asocian a los elementos de un menú.**
* **Editor de recursos: permiten gestionar los recursos que intervengan en la aplicación, como archivos con imágenes, por ejemplo.**

**4.3.- Descripción del problema.**

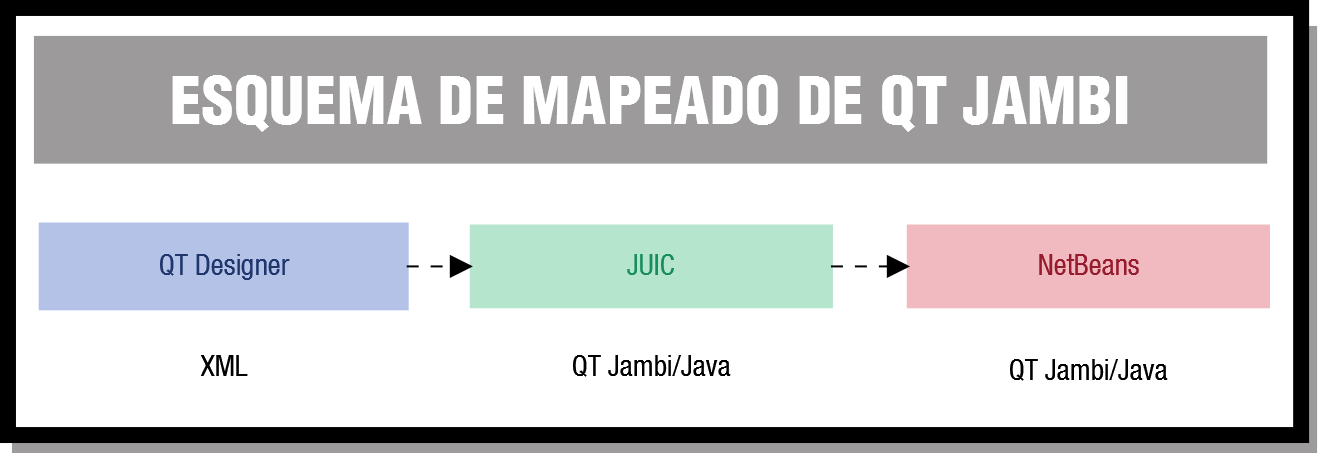
**En primer lugar, hay que destacar que en estos materiales se pretende dar un vistazo rápido al funcionamiento de una herramienta para la creación de interfaces de usuario usando XML, así como la posterior integración de esta interfaz en una aplicación. Por lo tanto no podrá considerarse esta unidad como un manual completo de aprendizaje de QT, sino como ejemplo didáctico de la materia que se pretende trasladar al alumnado.**

**Para ilustrar el uso de la herramienta Designer y la inclusión de los archivos de interfaz generados en una aplicación Java vamos a desarrollar una pequeña parte de una aplicación de gestión de un hotel, en concreto, haremos una interfaz muy sencilla para añadir una reserva de un cliente. Tendremos que escribir los datos personales del cliente y los datos de la reserva, que incluyen la fecha de entrada y salida, número de personas que se van a alojar en el hotel, tipo de habitación, pudiendo elegir entre doble de uso individual, doble, junior suite y suite, y si la habitación será para fumadores o no fumadores. En caso de que se seleccione una habitación para fumadores aparecerá un mensaje indicando que no está permitido fumar en ninguna otra zona del hotel.**

**El ejemplo se ha adaptado para poder incluir un mayor número de widgets en la interfaz y observar su funcionamiento. Esto nos permitirá usar los controles más básicos, como cuadros de texto, listas desplegables, o botones, y también alguno más elaborado como los calendarios. Para completar un poco el ejemplo crearemos una ventana principal extremadamente sencilla que permita ver como abrir esta interfaz desde un menú.**

**Posteriormente trataremos los archivos generados para que se pueda incluir en una aplicación java que estemos desarrollando con NetBeans, cada herramienta producirá un tipo de fichero diferente:**

**No vamos a implementar la funcionalidad subyacente a la interfaz, es decir, añadir la reserva a la base de datos del hotel, esto se deja para otros módulos del ciclo.**

****

**Reflexiona**

**Las interfaces de ejemplo son muy genéricas. Ten en cuenta que según el tratamiento que demos finalmente a la interfaz podremos obtener una aplicación de escritorio tradicional, una aplicación web o una aplicación para móviles. Sin embargo la interfaz a generar será siempre la misma y podremos usar el mismo archivo XML en los tres casos.**

**Es conveniente destacar que se ha adaptado y concretado mucho el problema con fines didácticos, no se pretende crear una aplicación funcional sino mostrar el funcionamiento de la herramienta mediante un ejemplo concreto que abarque todos los aspectos del proceso de creación de una interfaz con las tecnologías QT Jambi y Netbeans.**

**Una vez que sabemos lo que queremos conseguir, ¿por dónde empezamos?...**

**4.4.- Creación del formulario.**

**Recuerda que el acceso a la herramienta era ejecutando el archivo designer.sh que se encontraba en el directorio raíz qtjambi:**

**usuario@equipo:~/qtjambi$ ./designer.sh**

**Para crear una nueva interfaz seleccionamos File >> New , o directamente pulsamos la combinación Ctrl+N. Inicialmente nos pregunta qué tipo de ventana queremos crear y nos da a elegir entre:**

* **Diálogo con botones abajo (Dialog with buttons Bottom).**
* **Diálogo con botones a la derecha (Dialog with Buttons Right).**
* **Diálogo sin botones (Dialog without Buttons).**
* **Ventana principal (Main window).**
* **Widget**
* **Los diálogos se emplean para crear interfaces para la realización de tareas concretas que impliquen comunicación entre el usuario y la aplicación. Suelen usar botones para permitir la interacción con el usuario y realizar acciones.**
* **La ventana principal se suele usar como interfaz inicial de una aplicación. Incluye un menú y una barra de estado. Se le pueden añadir barras de herramientas.**
* **La interfaz widget se usa para extender la biblioteca y crear nuevos elementos de interfaz que podrán usarse posteriormente.**

**Una vez seleccionado el tipo de interfaz se van añadiendo los elementos de la misma seleccionándolos en el Widget Box y arrastrándolos a la zona de la interfaz donde se van a localizar.**

**En nuestro ejemplo crearemos una interfaz de reserva de habitación a partir de un diálogo sin botones. Se utilizará esta opción para ilustrar cómo se programan los botones después. También crearemos una ventana principal con un menú que permita abrir este diálogo.**

**Ambas interfaces se almacenan en formato XML en sendos archivos de texto plano, con extensión .jui, que podemos almacenar en un directorio de nuestro disco duro. Mas tarde, abriendo estos archivos, podemos recuperar las interfaces para seguir modificándolas desde la aplicación QT Designer o desde un editor de textos.**

**En los siguientes documentos podrás repasar como se añaden los elementos de interfaz al diálogo y a la ventana. Te recomiendo que los leas una vez para observar qué se pretende conseguir y cómo hacerlo. Luego puedes ir haciendo tú la interfaz al mismo tiempo que ves cómo se hace: "Anexo III.- Creación del diálogo para el alta de clientes. Anexo IV.- Creación del diálogo para el alta de clientes".**

**Para crear la ventana principal necesitarás la siguiente imagen:**

[**Imagen del hotel a incluir en la interfaz.**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R29_hotel.jpg) **(38 KB)**

**Autoevaluación**

Pregunta

En el proceso de construcción de la interfaz...

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-55_63): Basta con usar widgets sencillos, como etiquetas y campos de texto.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-55_76): Podremos insertar un menú en un cuadro de diálogo.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-55_79): Tenemos que tener mucho cuidado de colocar los widgets alineados.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-55_82): Arrestaremos los widgets a la interfaz sin atender excesivamente a su posicionamiento en la interfaz, ya que esto lo controlaremos después con el uso de layouts.

**Respuesta**: Opcion4. **Correcto, el uso de layouts permite gestionar el posicionamiento de los elementos de la interfaz según un diseño preestablecido.**

**4.5.- Modificar la composición de los elementos de la interfaz.**

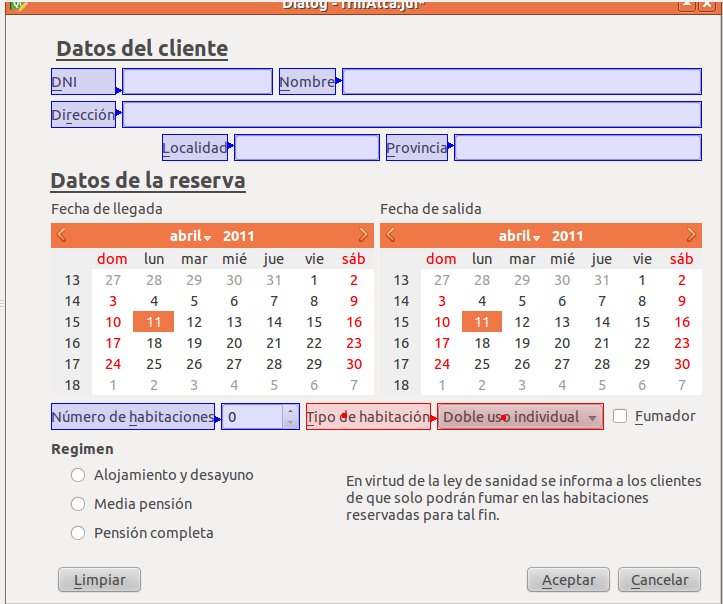
**Una vez que hemos situado los elementos que nos interesan vamos a modificar alguna de sus características. Principalmente haremos esto cambiando las propiedades de los objetos en el Editor de propiedades, pero también podemos cambiar algunas características haciendo clic con el botón secundario sobre el elemento a modificar. En concreto:**

* **Cambiaremos el nombre de los objetos para poder utilizarlos en el código más fácilmente. En principio vamos a dejar el nombre del tipo de objeto y lo acompañaremos de lo que representa. Por ejemplo, para el texto del nombre del cliente dejaremos el nombre del objeto, que es LineEdit, y lo acompañaremos de Nombre para saber de que objeto estamos hablando, quedando LineEdit\_Nombre.**
* **Modificaremos el texto de las etiquetas para asociarles una combinación de teclas. Por ejemplo, escribiremos en la etiqueta para el DNI "&DNI", esto lleva aparejado que al visualizar la interfaz aparezca la D subrayada de la siguiente manera: DNI. Veremos como hacer que al pulsar la combinación de teclas Alt+D el cursor se coloque automáticamente sobre el campo de texto para introducir el DNI.**
* **Añadiremos un ToolTipText. Es un pequeño texto que aparece al dejar el ratón sobre un control cierto tiempo para informar sobre su función, o el dato que debe introducirse.**
* **Añadiremos los elementos a la lista desplegable del tipo de habitación.**

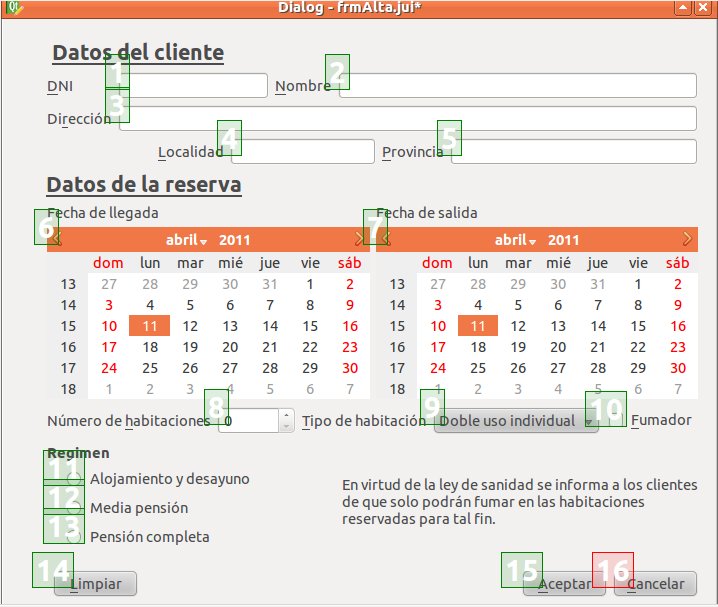
**Continuaremos el proceso de modificar los elementos de la interfaz haciendo que al pulsar la combinación de teclas de una etiqueta el foco recaiga sobre el control asociado. Para poder realizar este proceso hay que cambiar el modo de edición de la herramienta a "Editar Buddies".**

**Por último estableceremos el orden en el que se va a pasar de un control a otro cuando pulsemos la tecla tabulador, al visualizar la interfaz el cursor se coloca por defecto sobre el primer control modificable del formulario. Podemos modificar el orden en el que se va pasar al siguiente control cambiando el modo de edición a "Edit Tab Order" y seleccionando con el ratón el orden preferido.**

**En el siguiente documento puedes observar cómo se realizan estos cambios en las interfaces generadas: Anexo V.- Modificar las características de los elementos de la interfaz.**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_editar_buddies_en_QT_designer.html)

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Editar_tabulaciones.html)

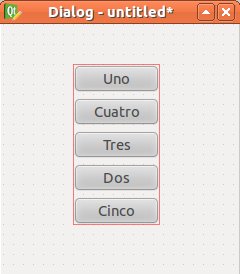
**4.6.- Definir la distribución de los elementos.**

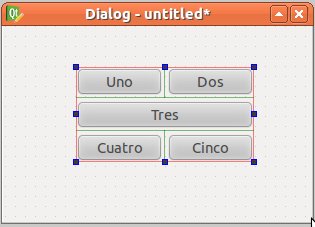
**Cuando se desarrolla una interfaz con QT Designer es preciso integrar sus elementos dentro de un layout para garantizar que al visualizarse los objetos se muestren en el lugar indicado. Así mismo nos aseguramos de que los objetos se dimensionarán adecuadamente cuando se redimensione el formulario.**

**Entendemos por layout o distribución el esquema que definirá la disposición de los elementos de la interfaz.**

**Podemos organizar los elementos como mejor nos convenga utilizando los siguientes tipos de diseño:**

* **Layout horizontal o vertical: es la distribución más sencilla. Encerrar un conjunto de elementos en un layout horizontal o vertical asegura que se van a mostrar alineados horizontal o verticalmente.**

****

* **Grid Layout: Permite una distribución compleja de los elementos de la interfaz ya que podemos colocar los objetos en una rejilla, sin embargo puede restar cierta flexibilidad al diseño.**
* **Splitter Layout: Permiten colocar los objetos con una distribución horizontal o vertical, pero dando la posibilidad de escoger el espacio final reservado para cada objeto.**

****

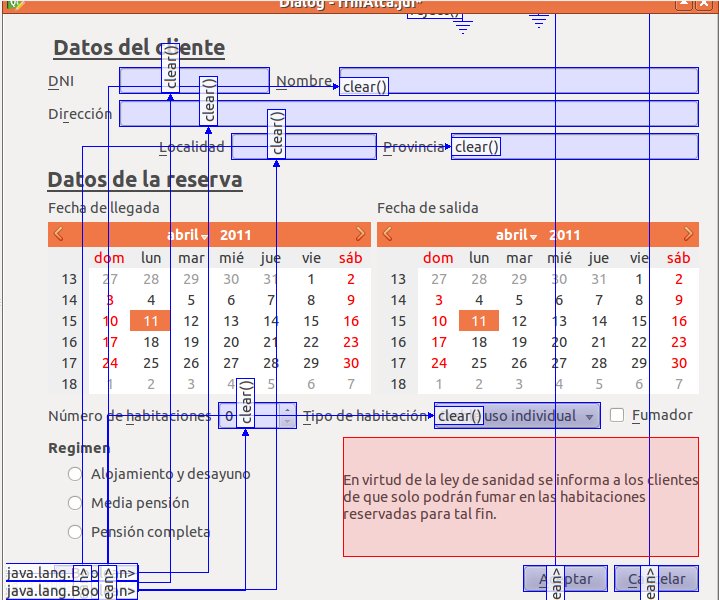
**En el siguiente documento veremos como definir el layout para los elementos de las interfaces que estamos desarrollando. Usaremos un Grid Layout para distribuir los datos del cliente, otro para las fechas de entrada y salida y otro para el resto de datos de la reserva. Los botones irán dentro de un Splitter Layout horizontal. "Anexo VI.- Añadir la distribución de los elementos de una interfaz."**

**4.7.- Añadir funcionalidad a los botones mediante conexiones signal-slot.**

**El mecanismo de Signal/slot, que literalmente significa señal-ranura, se emplea en QT Jambi para establecer comunicación entre los objetos de la interfaz. Por ejemplo, lo habitual es que al hacer clic en el botón Cerrar la ventana a la que pertenece se cierre, para conseguirlo asociamos a la señal “Hacer click” del botón con la ranura asociada a la función de cierre de la ventana.**

**Veamos con detenimiento estos conceptos:**

* **Una señal se produce cuando el objeto emisor cambia de estado de forma que pueda ser interesante para otros objetos. Los elementos QT Jambi poseen gran cantidad de señales predefinidas, pero es posible añadir señales a un objeto de ser necesario.**
* **Una ranura es un método que es llamado cuando se emite una señal. A pesar de existir gran cantidad de ranuras predefinidas es muy común que el programador escriba las suyas propias, como veremos más adelante.**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Editar_conexiones_signalslot.html)

**Este mecanismo es independiente del tipo, cuando se conecta una señal con una ranura, nos aseguramos de que al emitirse la señal se ejecutará el método asociado a la ranura, sin que el objeto emisor necesite saber nada del objeto sobre el que se ejecuta.**

**Es posible conectar tantas señales como sean oportunas a la misma ranura y conectar una señal a tantas ranuras como sea necesario.**

**Para crear las asociaciones entre señales y ranuras usaremos el modo de edición "Edit Signals/Slots" de la herramienta QT Designer. Haremos clic sobre el objeto emisor y arrastraremos hasta el objeto que recibe la señal, entonces veremos un diálogo que nos permita elegir entre las diferentes señales y ranuras que podemos escoger para estos objetos. En función de lo que seleccionemos se realizarán determinadas acciones ante la ejecución del usuario.**

**A continuación tienes un documento que ilustra este proceso: "Anexo VII.- Añadir funcionalidad a la interfaz."**

**Reflexiona**

**Al usar este mecanismo para gestionar la actividad de la interfaz conseguimos adicionar funcionalidad a la misma sin necesidad de escribir ni una sola línea de código, facilitando considerablemente la tarea del programador o programadora, y minimizando la aparición de errores de código.**

**Ahora tenemos la interfaz completa, pero una vez que la tenemos diseñada, ¿Cómo la integramos en una aplicación Java?**

**4.8.- El fichero .jui.**

**Una vez que tenemos el trabajo terminado usamos el menú File >> Save as para guardarlo en el disco duro y poder recuperarlo a posteriori. También podemos usar el icono con el acceso directo de la barra de herramientas.**

**Al guardar una interfaz generada con QT Designer obtenemos un archivo con extensión .<span style="font-size: small">jui</span>. En él se almacena la interfaz utilizando un lenguaje declarativo basado en XML. Cada elemento de la interfaz se representa como un elemento XML, y cumple con las características que vimos al principio de la unidad.**

**En el siguiente enlace podrás ver el código de las interfaces que hemos generado en el ejemplo anterior, te recomiendo abrirlo en otra pestaña del navegador para consultarlo mientras lees el resto del apartado.**

[**Código de la interfaz.**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R40_interfaces.tar.gz) **(898 KB)**

**Todos los elementos son widgets, para indicar el tipo de objeto se usa el atributo class, como puedes ver corresponde con cosas del tipo Qlabel, QlineEdit o QVBoxLayout, que también son nombres de clases de las bibliotecas QT.**

**Las propiedades de cada elemento o widget corresponden a subelementos XML denominados property, con el atributo name indicamos el nombre de la propiedad. Por ejemplo, para la etiqueta con el texto de la norma sobre fumar en lugares públicos tendremos un widget de clase Qlabel, y nombre label\_Norma, el texto aparece entre subelementos property con el nombre "text" y el texto dentro de una etiqueta string, y como quiero que se ajuste al ancho del control añado la propiedad "wordwrap" y la establezco a true mediante la subetiqueta bool.**

**<widget class="QLabel" name="label\_Norma">**

**<property name="text">**

**<string>En virtud de la ley de sanidad se informa a los clientes de que solo podrán fumar en las habitaciones reservadas para tal fin.</string>**

**</property>**

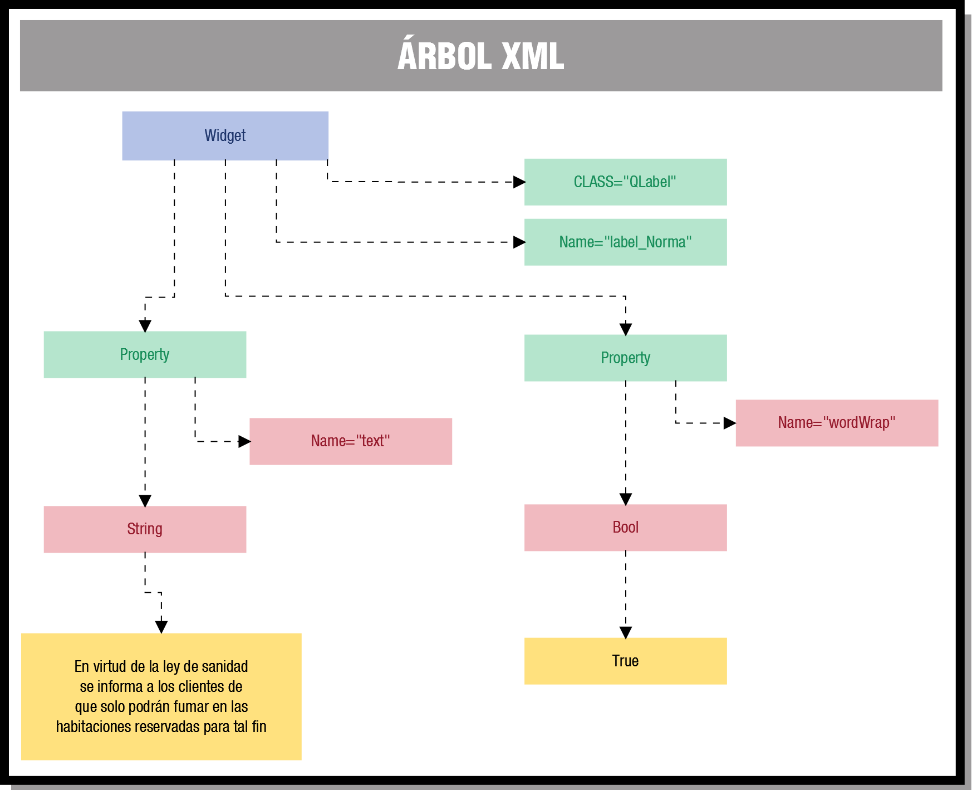
**<property name="wordWrap">**

**<bool>true</bool>**

**</property>**

**</widget>**

**Cualquier cambio que hagamos en este fichero con un editor de textos modificará la interfaz igual que si lo hiciéramos con la herramienta QT Designer.**

****

**Además de todos los elementos gráficos en el fichero .jui también se guardan los datos de las parejas signal/slot. Cada una de ellas encerrada entre <connection> y </connection> , ya que como veremos a continuación para asociar una señal a una ranura se realiza una conexión entre ellas.**

**Autoevaluación**

Pregunta

Internamente una interfaz generada con la herramienta QT Designer es...

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-71_63): Un archivo de texto plano que podemos integrar directamente en el proyecto de aplicación que estemos realizando.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-71_143): Un archivo binario que contiene los elementos de la interfaz mapeados a clases QT Jambi.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-71_146): Un archivo de texto plano que contiene la descripción de los elementos de la interfaz en formato XML.

[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-71_149): Un archivo de texto plano que contiene la descripción de los elementos estáticos y y ciertos aspectos dinámicos de la interfaz usando un lenguaje de descripción basado en XML.

**Respuesta**: Opcion4. **Correcto, es importante destacar que en este archivo además de los aspectos estáticos de la interfaz, como los elementos que la forman y su situación en la interfaz, se incluyen ciertos elementos de funcionalidad mediante las señales y ranuras predefinidas que podemos usar.**

**4.9.- Integrar la interfaz en una aplicación java.**

**Hasta el momento hemos estado generando interfaces de un modo sencillo y rápido, con la herramienta que estamos estudiando hemos podido colocar elementos de interfaz relativamente complejos y distribuirlos según unos criterios predefinidos, que aprovechan al máximo el espacio, y permiten redimensionar la interfaz con seguridad. Incluso se ha podido añadir cierta funcionalidad a todo el conjunto. Sin embargo, lo que hemos obtenido son dos archivo de extensión .jui, con un formato que no podemos integrar con el trabajo que estamos realizando en el todo el módulo y lo visto en la unidad anterior.**

**¿Cómo podemos añadir los ejemplos que hemos realizado a un proyecto java creado con NetBeans?, en principio no podremos hacerlo directamente, para ello tendremos que realizar una pequeña transformación sobre los archivos XML y añadir un par de bibliotecas a nuestro proyecto. Veamoslo en tres pasos:**

**Paso 1. Transformar las interfaces a código java.**

**Para ello vamos a usar la herramienta juic, que convierte archivos .jui en código fuente java. Podemos localizar la herramienta en el directorio bin dentro del directorio qtjambi. Si tenemos las interfaces en un directorio llamado "interfaces" dentro de qtjambi ejecutaremos lo siguiente:**

**usuario@equipo:~/qtjambi/interfaces/$ ~/qtjambi/bin/juic -pf frmAlta.jui**

**usuario@equipo:~/qtjambi/interfaces/$ ~/qtjambi/bin/juic -pf frmPrincipal.jui**

**El parámetro -pf elimina el prefijo del archivo generado que por defecto es Ui. Si la interfaz es un dialogo se genera automáticamente el archivo Dialog.java que contiene una clase llamada Dialog. Es una buena práctica cambiar el nombre del archivo generado a frmAlta.java y cambiar el nombre de la clase y el constructor una vez se edite con NetBeans.**

**De igual manera si la interfaz es una ventana se genera un archivo llamado MainWindow.java con su clase MainWindow, que también podemos modificar a frmPrincipal.java sin problemas.**

**Cabe destacar que si modificamos el archivo .java los cambios que hagamos se perderán si regeneramos el código fuente con juic.**

**Debes conocer**

**La herramienta juic tiene una serie de opciones que es importante que conozcas y que puedes encontrar en el siguiente documento.**

**"Anexo VIII.- Principales opciones de juic."**

**Paso 2. Crear el proyecto y añadir las clases con las interfaces al paquete.**

**En segundo lugar crearemos un proyecto nuevo con NetBeans, aunque no es imprescindible, podemos añadir las clases a un proyecto ya existente.**

**La incorporación de nuevos archivos fuente a un proyecto NetBeans es tan sencilla como copiar los archivos .java al directorio cuyo nombre coincide con el nombre del paquete por defecto del proyecto dentro del directorio src del proyecto en NetBeans.**

**Para que las clases formen parte efectivamente del paquete por defecto de la aplicación basta con añadir la sentencia package al principio del archivo.**

**En este momento debemos cambiar el nombre de la clase y el constructor a frmAlta y frmPrincipal.**

**No obstante lo que hemos hecho hasta el momento se detectan bastantes errores en el código. Esto es debido a que para definir los elementos gráficos de la interfaz empleamos una serie de clases cuya definición aún no encontramos. Para resolver este problema tendremos que añadir las bibliotecas que contienen estas definiciones.**

**Paso 3. Conectar las interfaces al método main.**

**Al crear el proyecto se genera de forma automática un archivo que contiene la clase Main, que a su vez contiene el imprescindible método main, punto de partida de la ejecución. Para poder conectar la interfaz a la aplicación tendremos que añadir el siguiente código al método main:**

**QApplication.initialize(args);**

**QMainWindow mw = new QMainWindow();**

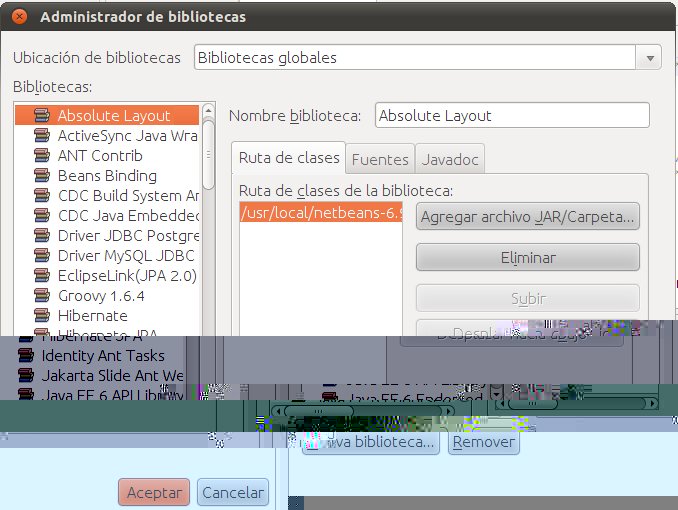
**frmPrincipal principal = new frmPrincipal();**

**principal.setupUi( mw );**

**mw.show();**

**QApplication.exec();**

**4.10.- Añadir las bibliotecas necesarias.**

****

**Para el correcto funcionamiento de la aplicación que se está desarrollando es preciso que tenga acceso a la definición de las clases QT que se emplean para implementar los elementos gráficos de la interfaz. Esta definición se incluye en las bibliotecas que se descargaron al principio desde la página de QT Jambi. Son cinco archivos jar que añadiremos a las bibliotecas de la herramienta NetBeans. Para ello hacemos clic en el menú Herramientas y a continuación en Bibliotecas.**

**Hacemos clic en Nueva Biblioteca. En la ventana que aparece ponemos de nombre a la nueva biblioteca qtjambi y aceptamos. A continuación podremos añadir los archivos .jar que descargamos desde el botón Agregar archivo JAR/Carpeta. Seleccionamos los cinco archivos .jar con las clases y ejemplos de qtjambi y los añadimos a la biblioteca qtjambi y pulsamos aceptar.**

**Pero esto no es suficiente para eliminar los errores en el código de la interfaz, ahora tenemos que hacer que las bibliotecas que hemos añadido se incluyan como parte del proyecto de ejemplo. Para ello nos aseguramos de que el proyecto está seleccionado en el panel de proyectos, desplegamos la pestaña de configuración del proyecto y seleccionamos Personalizar. Elegimos la categoría Bibliotecas, en principio no hay ninguna biblioteca seleccionada en la pestaña Compilar, desde aquí pinchamos en Añadir biblioteca, dentro de las bibliotecas para importar debe encontrarse disponible la biblioteca qtjambi que hemos añadido antes, pulsamos Añadir biblioteca y Aceptar.**

**Ahora se dispone de las definiciones de las clases utilizadas en la interfaz.**

**Autoevaluación**

Pregunta

Inserción de las bibliotecas jar de QTJambi en el proyecto...

Respuestas

[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-77_63): ...es algo innecesario.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-77_154): ...es suficiente con insertarlas en el administrador de bibliotecas de NetBeans.

[Opción 3](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-77_157): ...es necesario porque contienen las clases con la definición de la interfaz generada con QT Designer.

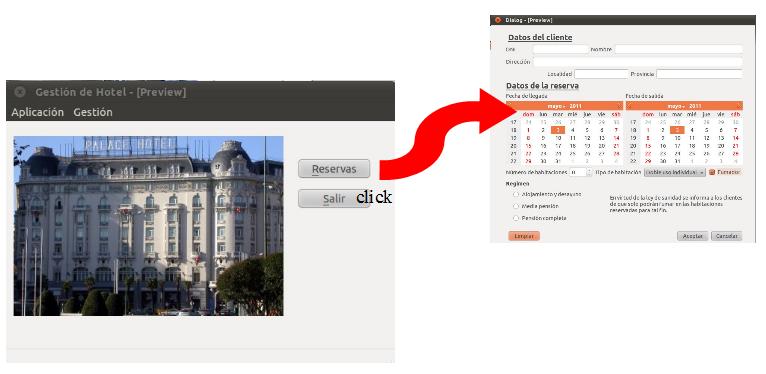
[Opción 4](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-77_160): ...es algo necesario porque permite acceder a la definición de las clases QT Jambi necesarias para la renderización de las interfaz generadas con QT Designer.

**Respuesta**: Opción4. **Correcto, estas bibliotecas contienen la base de QT Jambi y son imprescindibles para poder usar las interfaces generadas con QT Designer.**

**4.11.- Añadir funcionalidad de usuario.**

**Una vez que la interfaz está insertada en la aplicación final es posible seguir modificándola para añadirle cierta funcionalidad que la complete, de hecho, al ejecutar la aplicación vemos que podemos realizar las acciones que hemos definido en el apartado 4.6 mediante el mecanismo de señales y ranuras. Sin embargo tal y como lo hemos llevado a cabo mediante la aplicación QT Designer este mecanismo está muy limitado y no adquiere su verdadera dimensión hasta que el programador crea sus propias señales y ranuras.**

**Una ranura o slot no es más que una función que se ejecuta cuando se emite la señal a la que está conectado. Para generar la ranura sólo tenemos que añadir una función miembro a la clase de la interfaz. Para hacer que las funciones miembro formen parte del conjunto signal-slot tendremos que conectar la función al elemento que genere la señal utilizando la función connect y la señal precisa como veremos en el ejemplo.**

****

**Como ejemplo de estas funciones tenemos la que se ejecuta al pulsar un botón que cierra una ventana, o la función que lleve a cabo las medidas necesarias para informar y recuperar el estado normal de los elementos de una interfaz, cuando al aumentar el valor de un spinbox superamos cierto límite y podemos ejecutar.**

**En total, en nuestro caso, tenemos dos ranuras programadas por el usuario, una en la ventana principal y otra en el dialogo Reservas, en cada caso necesitaremos añadir un método que conectar al objeto y señal adecuados:**

1. **En la ventana principal al hacer clic sobre el botón Reservas se genera una instancia del cuadro de dialogo y se muestra. La señal se emite cuando hacemos clic en el botón.**
2. **En el dialogo para las reservas al hacer clic sobre la casilla de verificación Fumador se debe mostrar el mensaje sobre la prohibición de fumar en las zonas comunes del hotel, es decir, la visibilidad del mensaje, que aparece en una etiqueta, depende del estado de la casilla de verificación.**

**Además es preciso conectar las funcionalidades de abrir el dialogo de reservas y salir de la aplicación en el menú que hemos creado a tal efecto. Es tan sencillo como conectar las acciones asociadas al menú con las ranuras apropiadas.**

**Debes conocer**

**La implementación de las ranuras de usuario consiste en programar las funciones que ejecutan la ranura y conectarlas a los elementos adecuados de la interfaz.**

**Tienes una descripción detallada del código que debes modificar para implementar estas ranuras de usuario en este texto de enlace:**

**"Anexo IX.- Añadir ranuras de usuario."**

**Si te has perdido programando el código tiene una copia de las clases con las ranuras de usuario y de la clase principal en este texto de enlace:**

[**Código de las interfaces con la funcionalidad añadida por el programador.**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_CONT_R47_hotel.tar.gz) **(900 KB)**

**Autoevaluación**

Pregunta

La tecnología de programación de QT no está verdaderamente completa sin la programación de las ranuras propias del proyecto.

Respuestas

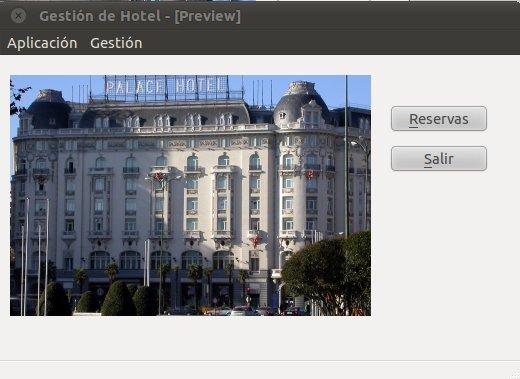
[Opción 1](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-84_63): Correcta.

[Opción 2](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/#answer-84_167): Falsa.

Respuesta: Opcion1. Correcta. **Correcto, el verdadero potencial de esta tecnología se alcanza cuando el programador incluye las funciones propias del proyecto y las asocia a la emisión de las señales correspondientes.**

**4.12.- Notas finales.**

**Hemos "terminado" la aplicación, cuya interfaz queda así:**

****

**Aún así para que el programa funcione adecuadamente hay que añadir un par de detalles:**

* **Es preciso que al cargar el formulario de reservas la etiqueta con la norma de sanidad aparezca oculta, por lo que añadimos en el código de la función setupUI la linea:**
* **label\_Norma.setVisible(false);**

**Que hace que, inicialmente, se muestre oculta.**

* **Para ver correctamente la imagen del hotel, hay que empaquetar la imagen en una archivo .jar e importarlo en la opción Personalizar >> Bibliotecas >> Añadir JAR/Carpeta. La sentencia**

**se sustituye por:**

**label.setPixmap(new Qpixmap(("classpath:recursos/hotel.jpg")));**

**Suponiendo que la imagen se encuentre en un directorio llamado "recursos" y que este se haya empaquetado también con la orden:**

**jar -cf recursos.jar recursos**

**Anexo I.- Instalar QT\_Jambi en Linux Ubuntu.**

**Nota: En este documento se describirá el proceso de descarga e instalación de la versión 4.7.0-beta2 para un sistema con Linux Ubuntu Desktop 10.10 instalado en una arquitectura de 64 bits, pero no existe una gran diferencia con otras arquitecturas o sistemas operativos.**

**Las bibliotecas QT-Jambi estan disponibles para las plataformas Linux, Windows y MAC-OS. Al abrir la página web seleccionamos el enlace que más nos convenga según nuestra arquitectura (32 o 64 bits).**

**El proceso de descarga nos deja un archivo denominado qt-jambi-linux64-community-4.7.0-beta2.tar.gz en carpeta Descargas que deberemos copiar a nuestro directorio principal (/home/nombre\_usuario habitualmente) y descomprimir:**

1. **Copiamos el archivo en el raíz de nuestro directorio personal.**

**usuario@equipo:~/Descargas$ cp qt-jambi-linux64-community-4.7.0-beta2.tar.gz ~**

1. **Descomprimimos los archivos, usando la siguiente orden:**

**usuario@equipo:~tar -xvf qtjambi-linux64-community-4.7.0-beta2.tar.gz**

1. **Ahora tenemos un directorio llamado qtjambi-linux64-community-4.7.0 con las bibliotecas y utilidades que vamos a necesitar para desarrollar nuestro trabajo, por motivos prácticos vamos a renombrar el directorio dejándolo solo como qtjambi:**

**usuario@equipo:~mv qtjambi-linux64-community-4.7.0 qtjambi**

1. **Para un correcto funcionamiento de las bibliotecas tendremos que establecer la variable de entorno JAVA\_HOME asignándole como valor el directorio donde se encuentra instalada la versión de Java que estemos usando en nuestro sistema, por ejemplo, podemos modificar el archivo .profile añadiendo como última línea:**

**$JAVA\_HOME=/usr/local/jdk1.6.0\_22 siendo /usr/local/jdk1.6.0\_22 el directorio donde están los archivos del entorno de desarrollo java.**

**Si observamos el contenido del directorio qtjambi encontraremos, entre otras cosas, lo siguiente:**

* **Un conjunto de archivos jar que tendremos que importar en nuestros proyectos para poder usar el binding.**
* **El archivo readme.html hace una pequeña presentación de QT-Jambi y tiene un enlace a la página con la documentación de la web oficial de QT.**
* **También vemos un archivo de proceso por lotes, qtjambi.sh, que al ejecutarlo muestra una serie de ejemplos desarrollados en QT-Jambi. Se puede acceder tanto al resultado final del ejemplo como al código necesario para llevarlo a cabo.**
* **El archivo designer.sh abre una versión de la herramienta QT Designer que usaremos para desarrollar las interfaces que serán generadas en código XML.**

**Anexo II.- Instalar QTJambi Windows 64.**

**Nota: La instalación en Windows 32 no debe dar ningún tipo de problema con las versiones disponibles en la página de descarga de QtJambi. Sí se pueden dar problemas al ejecutar QtJambi en Windows 64. Vamos a explicar la descarga e instalación con una versión verificada.**

**Iremos a la página de descarga de QtJambi:**[**https://sourceforge.net/projects/qtjambi/**](https://sourceforge.net/projects/qtjambi/)**. En este caso podemos descargarnos las versiones que aparecen como actuales: Versiones 4.8.6 y 4.8.7 para Windows 32 y 64 respectivamente.**

**Descargamos, descomprimimos e instalamos la versión seleccionada.**

**En el caso de que se generen errores al ejecutar QtJambi (más corriente en sistemas operativos Windows 64) procederemos de la siguiente forma:**

1. **En la página de descarga antes citada seleccionaremos "For other official releases check our SourceForge page."**
2. **Nos direccionará a la página *"http://sourceforge.net/projects/qtjambi/files/"*.**
3. **En la tabla de versiones que se muestra seleccionaremos para su descarga la versión 4.5.2(última posición de la tabla).**
4. **Accedemos a *"http://sourceforge.net/projects/qtjambi/files/4.5.2/".***
5. **Descargaremos la versión de 64 bits para windows qtjambi-win64-lgpl-4.5.2\_01.zip.**
6. **Finalmente debemos descomprimir e instalar en un directorio particular.**

**Nota: En el caso de que ya hayamos instalado alguna otra versión es conveniente desinstalarla previamente. En el caso que se desee instalar esta versión para 32 bits en la tabla de versiones descrita en el punto 3 deberemos seleccionar la versión qtjambi-win32-lgpl-4.5.2\_01.zip.**

**Anexo III.- Creación del diálogo para el alta de clientes.**

**Requisitos de la interfaz.**

**Se pretende crear una interfaz para gestionar las reservas de un hotel. Tenemos que poder escribir los datos personales de la persona que hace la reserva:**

* **DNI.**
* **Nombre.**
* **Dirección.**
* **Localidad.**
* **Provincia.**

**También tenemos que poder incluir los datos de la reserva:**

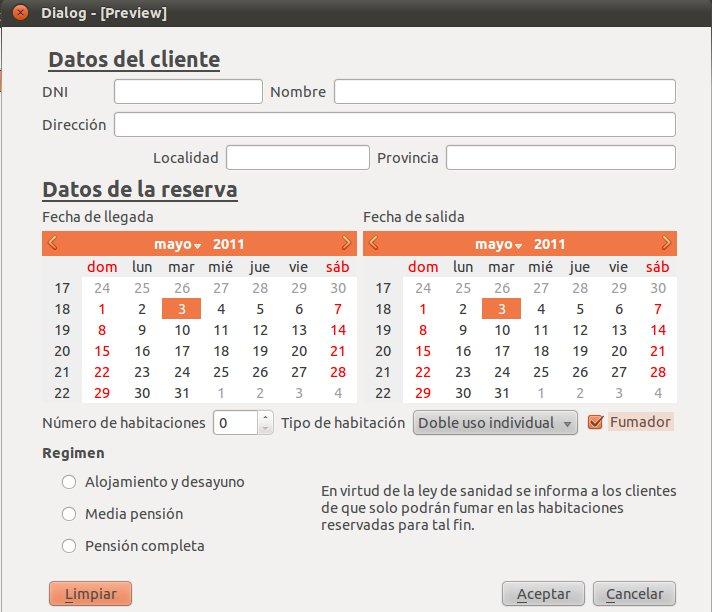
* **Fechas de llegada y salida.**
* **Número de personas.**
* **Tipo de habitación. El tipo de habitación puede ser:** 
  + **Doble de uso individual.**
  + **Doble.**
  + **Junior suite.**
  + **Suite.**
* **Si es para fumador o no fumador.**
* **Régimen de alojamiento. Puede ser:** 
  + **alojamiento y desayuno.**
  + **Media pensión.**
  + **Pensión completa.**

**Como requerimiento adicional se pide que si la reserva es para fumador se muestre el siguiente mensaje:**

**"En virtud de la ley de sanidad se informa a los clientes de que solo podrán fumar en las habitaciones reservadas para tal fin."**

**La interfaz debe contar con las facilidades de atajos de teclado y manejo del tabulador.**

**Una vez terminada se verá así:**

****

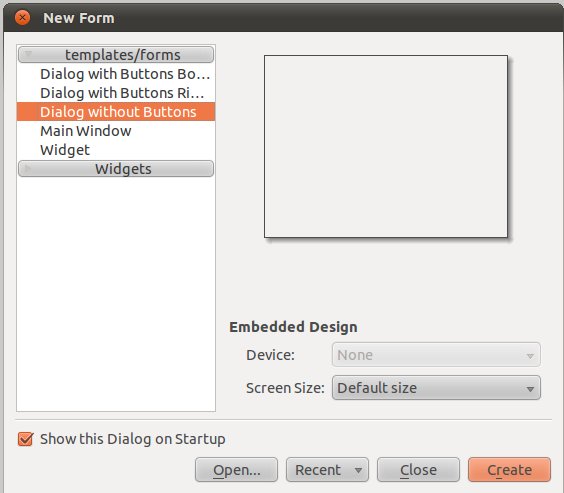
**Elaboración de la interfaz.**

**Arrancamos QT Designer:**

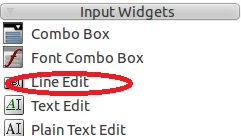
**usuario@equipo:~/qtjambi/bin$ ./designer.sh**

**Para empezar nos pide que seleccionemos el tipo de interfaz, en nuestro caso, vamos a crear un diálogo, podemos utilizar el modelo sin botones en la zona inferior.**

**Como tiene los nombres en inglés luego les cambiaremos las etiquetas.**

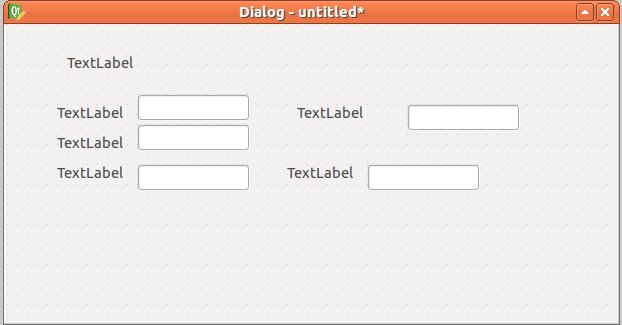
****

**El siguiente paso es añadir los widgets que compondrán la interfaz. En esta herramienta hay que arrastrar el widget desde el panel de la izquierda, hasta situarlo en la zona que queremos. Para insertar los datos del cliente necesitamos etiquetas (Labels) y campos de texto (LineEdits).**

****

****

**La base para los datos del cliente queda más o menos así:**

****

**Si hacemos doble clic en una etiqueta podemos editar el texto que aparece, modificamos las que tenemos para que queden como en la imagen de la derecha. Es importante dejar los símbolos & para la edición de buddies, como veremos más adelante.**

**Podemos modificar el formato de una etiqueta en el panel de propiedades, buscamos la propiedad font.**

****

**Modificamos el formato de fuente a Ubuntu, negrita, tamaño 16, quedando la interfaz así:**

****

**Pasamos ahora a añadir los widgets para los datos de la reserva.**

* **Fechas de llegada y salida >> Calendar.**
* **Número de personas >> Spinbox.**
* **Tipo de habitación >> ComboBox.**
* **Si es para fumador o no fumador >> Checkbox.**
* **Régimen de alojamiento >> Botones de radio (Radio Button) dentro de un GroupBox.**
* **El texto con el aviso sobre la prohibición de fumar en el hotel >> Label**
* **Todos los nombres son etiquetas (Label)**

**Para que quepan todos tendremos que agrandar la ventana del diálogo, se agranda arrastrando con el ratón desde los bordes de la ventana. Para mover los botones basta con seleccionarlos con el ratón y arrastrarlos.**

**Nota: Cuando añadas el GroupBox y los botones de radio arrastra primero el GroupBox y los botones de radio después para que queden dentro del grupo.**

**La interfaz queda así:**

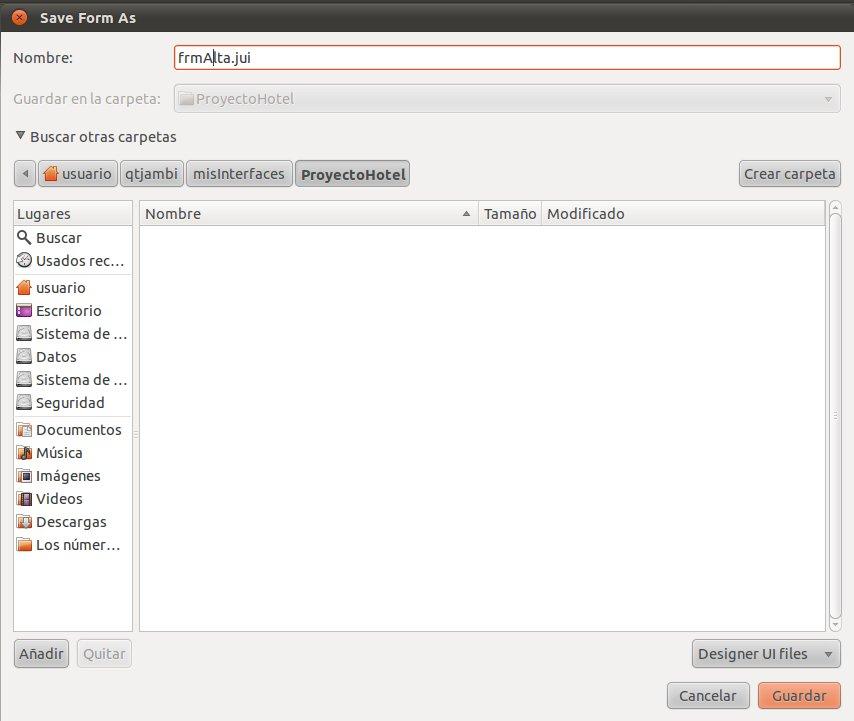
****

**Ahora modificamos el texto de las etiquetas haciendo doble clic sobre ellas, para que queden como en la imagen ,y modificamos, también, el formato de la fuente de la etiqueta "Datos de la reserva" a Ubuntu, 16, negrita, de la misma forma que lo hicimos con "Datos del cliente".**

**Añadiremos un botón que falta para reiniciar todos los widgets de la interfaz, si es necesario. Para ello añadimos un PushButton, haciendo doble clic sobre el podemos cambiar el texto que aparece a &Limpiar. Y dos más para Aceptar y Cancelar y ya tenemos la interfaz diseñada.**

****

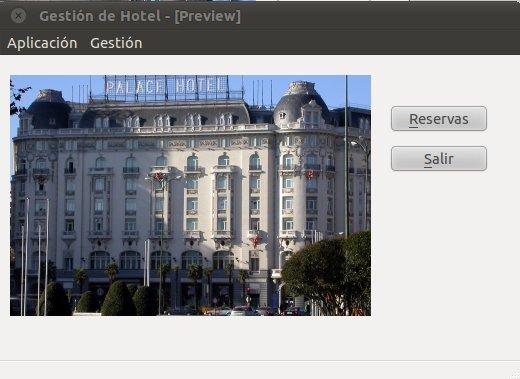
**Ahora guardamos la interfaz en un directorio que tengamos preparado para ello, por ejemplo, en nuestro caso hemos creado el directorio ProyectoHotel dentro de misInterfaces dentro de qtjambi. De esta forma tendremos un directorio para cada proyecto que hagamos con QT Designer.Para guardar una interfaz basta con hacer clic en File >> Save as... y obtendremos el siguiente cuadro de dialogo:**

****

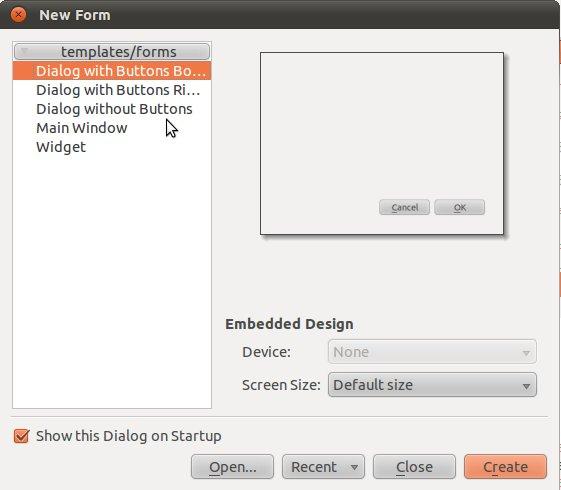
**Anexo IV.- Creación de la interfaz principal usando QT Designer.**

**Requisitos de la interfaz.**

**En este segundo documento crearemos la interfaz para la ventana principal de la aplicación de gestión del hotel. Por motivos de tiempo, pero sobre todo didácticos se va a generar una interfaz muy sencilla y limitada, y tan solo se van a generar elementos como botones y menús para el diálogo que hemos creado. La interfaz debe contar con las facilidades de atajos de teclado y manejo del tabulador.**

**Una vez terminada se verá así: **

**Elaboración de la interfaz.**

****

**Arrancamos QT Designer:**

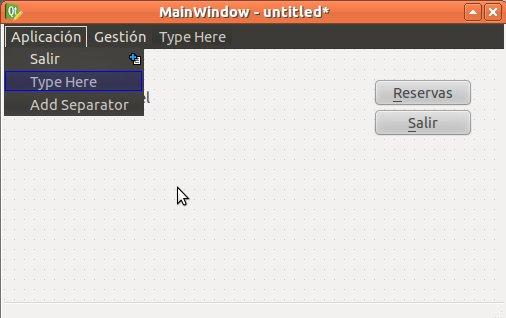
**usuario@equipo:~/qtjambi$**

**./designer.sh**

**Para empezar nos pide que seleccionemos el tipo de interfaz, en nuestro caso, vamos a crear una ventana principal (Main Window).**

**Al crear una interfaz de tipo ventana principal, advertimos, como principal diferencia con el diálogo, que incluye facilidades para crear un menú, además del lienzo donde añadir los widgets. Esta interfaz será mucho más sencilla que la anterior, a continuación los pasos para crearla:**

1. **Disminuiremos su tamaño para hacerla más pequeña, a la derecha aparece el inspector de propiedades en donde podemos encontrar la propiedad Geometry, si desplegamos la pequeña flecha que aparece a la izquierda veremos los parámetros que forman la geometría. En width y height ponemos los valores 500 y 300.**
2. **Añadiremos una etiqueta (Label) y dos botones (PushButton). Haciendo doble clic sobre cada botón podemos modificar su nombre a &Reservas y &Salir.**
3. **Haciendo doble clic sobre el texto "Type Here" de los elementos del menú podemos modificar el texto que aparece, cada vez que modificamos una entrada de menú y aceptamos obtenemos nuevos elementos de menú para modificar:**

****

**El menú Aplicación tiene un único elemento: “Salir” y el menú Gestión tiene como único elemento “Reservas”, que serán las dos acciones principales que se realizarán desde esta interfaz: Salir de la aplicación y abrir el diálogo reservas.**

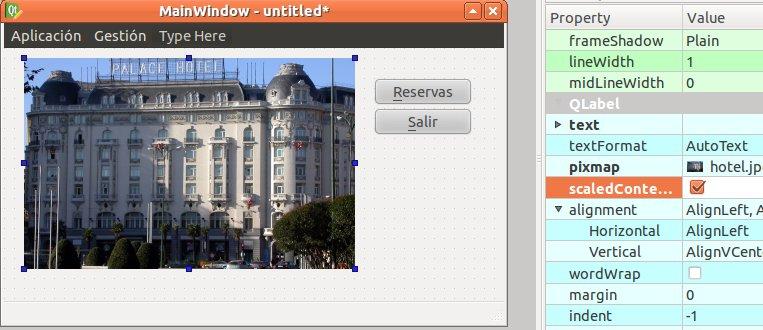
1. **Guardamos la interfaz como “frmPrincipal.jui”.**

**Insertar una imagen en la interfaz.**

**Para añadir imágenes en las interfaces QT usamos etiquetas. En concreto en nuestro ejemplo hemos añadido una foto de la fachada de un hotel.**

**Almacenamos la imagen del hotel en un directorio, llamado, por ejemplo, "recursos" en donde tengamos las interfaces .jui.**

**Para que aparezca seleccionamos la etiqueta que hemos añadido antes y vamos al inspector de propiedades. Seleccionamos la propiedad "pixmap", que pertenece a la categoría QLabel. A la derecha aparece una flechita, si la pulsamos nos permite elegir entre "Choose Resource..." y "Choose File...", elegimos esta segunda opción. Aparecerá un cuadro de diálogo para seleccionar un archivo, si seleccionamos el archivo con la imagen veremos como, automáticamente, ésta se carga en la etiqueta. Para poder bien la imagen tendremos que marcar la casilla de verificación "scaledContent".**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Anadir_imagen_a_la_interfaz_principal.html)

**Anexo V.- Modificar los atributos de una interfaz gráfica usando QT Designer.**

**Requisitos de la interfaz.**

**En el punto anterior generamos esta interfaz para gestionar, en un modo muy simple, las reservas de un hotel:**

**Creamos un cuadro de diálogo y añadimos los elementos gráficos de interfaz, hemos modificado algunas características básicas como los textos asociados que aparecen en la interfaz, ahora vamos a continuar haciendo cambios de algunas propiedades.**

**Propiedades comunes.**

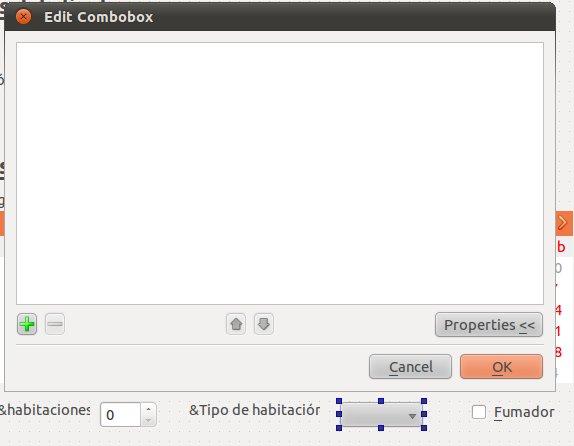
**Primero vamos a cambiar las propiedades comunes a todos los elementos. Si hacemos clic con el botón secundario sobre un elemento vemos varias opciones, entre ellas:**

* **Change objetcName... >> con esto cambiamos el nombre al objeto. Al cambiar el nombre del objeto podremos identificarlo más fácilmente después en el código. En nuestro caso dejamos como nombre NombreObjeto\_dato, siendo dato la etiqueta que acompaña al elemento, por ejemplo, el campo de texto Nombre lo guardamos como LineEdit\_Nombre.**
* **Chage ToolTipText... >> un TooltipText es un cuadro, normalmente amarillo, que contiene cierta información que puede ser de utilidad para el usuario de la interfaz, como que tipo de dato hay que insertar o su formato. Aparece una vez que dejamos el ratón sobre el elemento cierto tiempo. Para el nombre podemos usar como ToolTipText "Nombre del cliente".**

**Propiedades de la lista desplegable (comboBox)**

**Utilizamos la lista desplegable o comboBox para poder seleccionar un elemento de una lista de posibles. Luego, antes de usar la interfaz, será preciso indicar qué elementos son esos.**

**Si hacemos doble clic sobre el elemento vemos una diálogo para crear esta lista:**

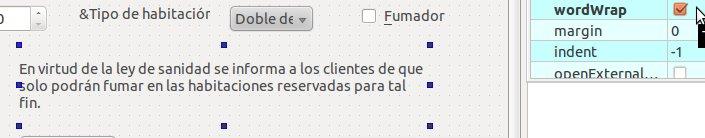
****

**Usando el botón + aparece una zona en la que podemos escribir el texto de cada nuevo elemento, que confirmamos pulsando Intro. Cuando tengamos la lista hecha podemos corregir errores haciendo doble clic sobre el elemento erróneo o modificar el orden con las flechas. Recuerda que la lista desplegable se rellena con las opciones:**

* **Doble de uso individual.**
* **Doble.**
* **Junior suite.**
* **Suite.**

**Propiedades de la etiqueta.**

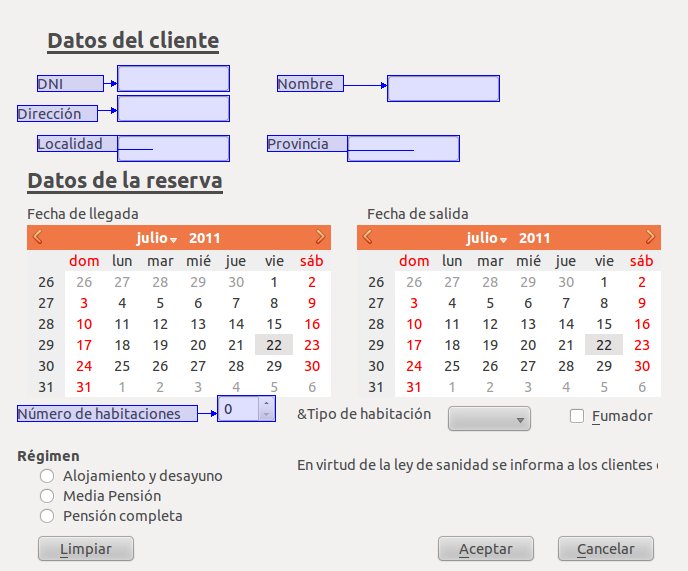
**La etiqueta con la norma sobre fumar en el hotel contiene un texto demasiado largo, por lo que no cabe en el espacio que le hemos reservado. Para cambiar esto seleccionamos la etiqueta y vamos al inspector de propiedades. Buscamos la propiedad wordwrap y la seleccionamos.**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Modificar_las_propiedad_de_salto_de_linea_de_una_etiqueta.html)

**Editar buddies**

**Continuaremos el proceso de modificar los elementos de la interfaz haciendo que al pulsar la combinación de teclas de una etiqueta el foco recaiga sobre el control asociado. Para poder realizar este proceso hay que cambiar el modo de edición de la herramienta a "Editar Buddies".**

****

**La asociación se realiza arrastrando desde la etiqueta hasta el elemento asociado.**

**Editar orden de tabulación**

****

**Por último estableceremos el orden en el que se va a pasar de un control a otro cuando pulsemos la tecla tabulador, al visualizar la interfaz el cursor se coloca por defecto sobre el primer control modificable del formulario. Podemos modificar el orden en el que se va pasar al siguiente control cambiando el modo de edición a "Edit Tab Order" y seleccionando con el ratón el orden preferido.**

**Anexo VI.- Añadir la distribución de los elementos de una interfaz.**

**Descripción de la interfaz.**

**En puntos anteriores hemos generamos esta interfaz para gestionar, en un modo muy simple, las reservas de un hotel:**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Diseno_de_interfaz_con_todos_los_elementos_Completa.html)

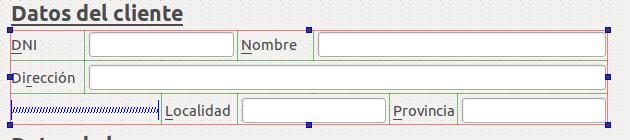
**Creamos un cuadro de diálogo y añadimos los elementos gráficos de formulario, hemos modificado algunas características básicas como los textos asociados que aparecen en la interfaz, los nombres de los objetos, y otras propiedades necesarias para completar su funcionamiento. También se han asociado las etiquetas a sus objetos para permitir los atajos de teclado y se ha establecido el orden de tabulación. Ahora vamos a ordenar y distribuir adecuadamente los objetos de esta interfaz.**

**Distribución de los datos del cliente.**

**Para este grupo de objetos vamos a usar un tipo de distribución denominada GridLayout que encaja los elementos en una rejilla bidimensional. Puesto que los elementos de la interfaz no forman una cuadrícula exacta usamos los widgets de la categoría Spacers para ocupar zonas concretas en la distribución. También modificamos el tamaño de los campos de texto para que se ajusten a lo que van a contener:**

****

**Seleccionamos todos los elementos de la interfaz, hacemos clic con el botón secundario y en la opción del menú contextual Layout elegimos "Layout in a Grid".**

****

**Distribución de las fechas**

**El grupo de calendarios y etiquetas de las fechas de llegada y salida también van en un GridLayout. Procedemos seleccionando los cuatro objetos y seleccionando en la opción del menú contextual contextual "Layout in a Grid".**

****

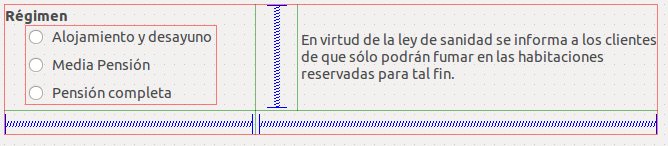
**Distribución para Número de habitaciones, Tipo y Fumador.**

**Estos elementos los colocaremos en layout horizontal. Seleccionamos los objetos, hacemos clic con el botón secundario y seleccionamos en Layout "Layout Horizontally".**

****

**Distribución para el resto de los elementos.**

**Para distribuir el régimen de alojamiento y la nota a los fumadores vamos a usar primero un layout vertical para los tres tipos de régimen, y después distribuimos el resto de elementos en un Grid Layout con espaciadores para asegurarnos de que los datos del régimen de alojamiento aparecen correctamente :**

****

**Distribución para los botones.**

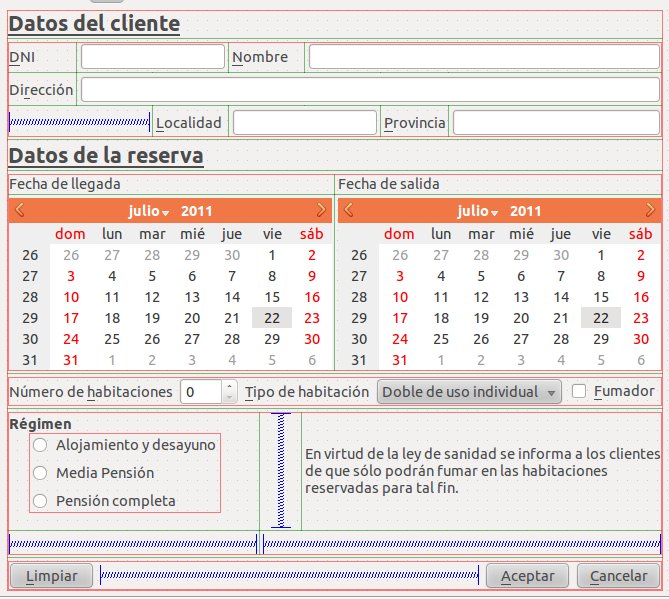
**Los botones se distribuyen con un layout horizontal, usaremos un espaciador para separar el botón de limpiar de los botones de Aceptar y Cancelar.**

****

**Distribución general.**

**Ahora establecemos la distribución general para toda la interfaz. Seleccionamos todos los elementos del formulario y le asignamos un GridLayout, quedando finalmente así:**

**En última instancia aplicamos la distribución al formulario, para lo cual hacemos clic en cualquier zona libre de la interfaz y seleccionamos la opción "Layout in a FormLayout". Con esto conseguimos ajustar los elementos del formulario a la esquina superior izquierda de la ventana y podremos ajustar su tamaño final.**

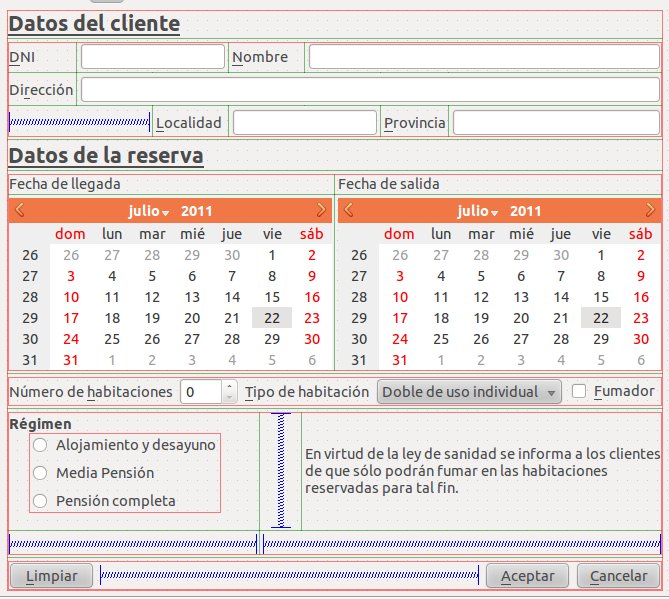
****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Distribucion__de_los_botones.html)

**Anexo VII.- Añadir funcionalidad a la interfaz.**

**Descripción de la interfaz.**

**En puntos anteriores hemos generamos esta interfaz para gestionar, en un modo muy simple, las reservas de un hotel:**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Distribucion__de_los_botones.html)

**Creamos un cuadro de diálogo y añadimos los elementos gráficos de interfaz, hemos modificado algunas características básicas como los textos asociados que aparecen en la interfaz, los nombres de los objetos, y otras propiedades necesarias para completar su funcionamiento. También se han asociado las etiquetas a sus objetos para permitir los atajos de teclado y se ha establecido el orden de tabulación. Se han ordenado y distribuido adecuadamente los objetos de esta interfaz mediante layouts.**

**Ahora vamos a hacer que, al producirse determinado acciones sobre los elementos de la interfaz se realicen las tareas que se indiquen.**

**Funcionalidad que se pretende añadir.**

* **Vamos a empezar con algo sencillo, los botones OK y Cancel, cierran ambos el diálogo, el valor devuelto depende de como se cierre.**
* **El botón Limpiar hacer que el resto de los elementos de la interfaz vuelvan a sus valores originales.**

**Para asignar estas acciones a los botones, tenemos que poner el editor en modo "Edit Signal/Slot":**

****

**Puesto que los botones OK y Cancel habían sido incluidos por defecto, también por defecto llevan asociada su funcionalidad.**

**Para el botón Limpiar haremos lo siguiente:**

Ventana de una herramienta con el título "Configure connection", distribuida en dos columnas, a la izquierda aparece el texto pushButton(QPushButton), debajo un recuadro grande de color blanco con los textos:

* clicked <java.lang.Boolen>
* pressed
* released
* toggled <java.lang.Boolean>

uno debajo del otro, y a continuación un rectángulo gris claro con el texto "Edit...".

En la columna de la izquierda aparece el texto lineEdit (QLineEdit), y debajo un recuadro blanco con los textos:

* clear()
* cut()
* paste()
* redo()
* selectAll()
* undo()

y a continuación un rectángulo gris claro con el texto "Edit...".

En la parte inferior de la ventana se aprecia un cuadro pequeño de color blanco y acompañándole el texto "Show signals and slots from Qwidget".

Debajo a la izquerda dos rectángulos grises con los textos Cancel y OK.

1. **Hacemos clic con el ratón sobre el botón y sin soltarlo lo arrastramos hasta el primer cuadro de texto: DNI.**
2. **Al soltar el ratón aparecerá una ventana como la siguiente: A la izquierda aparece el objeto sobre el que se emite la señal y a la derecha el objeto sobre el que recae la acción (también puede ser el dialogo). En nuestro caso el emisor es un pushButton, con la lista de señales que se ve en la imagen y el receptor un cuadro de texto, con las acciones que puede ejecutar. Cuando pulsemos el botón (clicked) el campo de texto se borrará (clear()). Si no encontramos la función que nos haga falta pulsamos en Show signals and slots inherit from QWidget para ver más señales y ranuras.**
3. **Repetimos el proceso para todos los elementos que pueden volver a su estado inicial de la interfaz: campos de texto, spinbox, lista desplegable, casilla de verificación y botones de radio. En todos los casos elegimos la señal clicked y la función clear().**

**Anexo VIII.- Principales opciones de juic.**

**Opciones de ejecución de la herramienta JUIC.**

* **-h, -help: describe cómo funciona y muestra una lista con las opciones del comando.**
* **-v, -version: muestra el número de versión del compilador.**
* **-d**

**: especifica el directorio de salida.**

* **-x : carga el archivo de configuración propio.**
* **-p : especifica un paquete para el archivo generado -class (es relativo al directorio de salida).**
* **-cp : actualiza todos los archivo .jui en el path proporcionado. El argumento path es opcional; el path por defecto es $CLASSPATH.**
* **-pf : especifica el prefijo utilizado para las clases generadas. JUIC añade el prefijo "Ui\_" de forma automática a las clases generadas. Utilice esta opción para modificar el prefijo, si se omite el argumento se eliminará el prefijo.**
* **-i :<...>:: usado en combinación con la opción 'cp' permite que JUIC actualice solo los archivos especificados.**
* **-e :<...>:: usado en combinación con la opción 'cp' impide que JUIC actualice los archivos especificados.**
* **-a: actualiza los archivos independientemente de la fecha de modificación.**

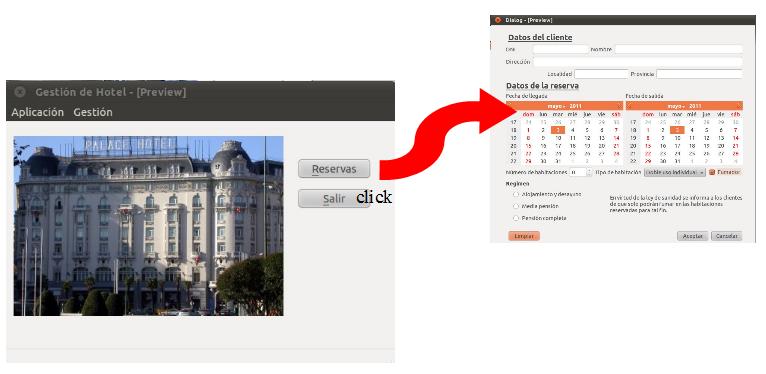
**Anexo IX.- Añadir ranuras de usuario.**

**En total, en nuestro caso, tenemos dos ranuras programadas por el usuario, una en la ventana principal y otra en el dialogo Reservas, en cada caso necesitaremos añadir un método que conectar al objeto y señal adecuados.**

**Abrir el dialogo de la reserva.**

1. **En la ventana principal al hacer clic sobre el botón Reservas se genera una instancia del cuadro de dialogo y se muestra. La señal se emite cuando hacemos clic en el botón.**

**La función a ejecutar debe crear un dialogo nuevo como esta:**

****

**void abrir(){**

**frmAlta Alta = new frmAlta();**

**QDialog dialog = new QDialog();**

**Alta.setupUi(dialog);**

**dialog.show();**

**}**

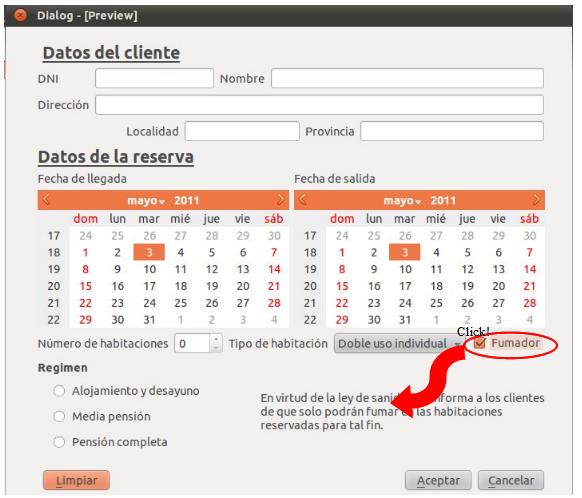
**Si el botón se llama pushButton\_Reservas conectaremos la función de la siguiente manera en el método setupUi:**

**pushButton\_Reservas.clicked.connect(this, "abrir()");**

**Esto quiere decir que cuando se emita la señal "clicked" (pulsado) sobre el botón se ejecutará la función abrir(). this hace referencia al elemento que recibe la acción a la ranura, en este caso la propia ventana.**

**Mostrar el mensaje para los fumadores.**

1. **En el dialogo para las reservas al hacer clic sobre la casilla de verificación Fumador se debe mostrar el mensaje sobre la prohibición de fumar en las zonas comunes del hotel, es decir, la visibilidad del mensaje, que aparece en una etiqueta, depende del estado de la casilla de verificación.**

****

[**Resumen textual alternativo**](https://educacionadistancia.juntadeandalucia.es/formacionprofesional/pluginfile.php/48943/mod_scorm/content/0/DI02_Descripcion_Imagen_Mostrar_etiqueta_copia.html)

**Para gestionar esto podemos crear un método que se llame cambiar() como el siguiente:**

**void cambiar(){**

**//si se ha activado la casilla de verificacion mostraremos el mensaje**

**//la visibilidad de la etiqueta**

**//depende de si el checkbox está activado**

**label\_Norma.setVisible(checkBox\_Fumador.isChecked());**

**}**

**Para conectar el método a la casilla de verificación, y más concretamente cuando se emita la señal de pulsado añadiremos la siguiente linea:**

**checkBox\_Fumador.clicked.connect(this, "cambiar()");**

**En la función setupUi, junto con el resto, creadas con QTDesigner.**

**Nota: los métodos con las ranauras de usuario se deben incluir antes del método setupUi de la clase, la conexión se realiza en el propio método setupUi, puede hacerse justo antes de llamar al método connectSlotsByName, al final.**

**Añadir funcionalidad al menú**

1. **Por último nos quedaría conectar las funcionalidades de abrir el dialogo de reservas y salir de la aplicación en el menú que hemos creado a tal efecto. Es tan sencillo como conectar las acciones asociadas al menú con las ranuras apropiadas. Cada entrada de menú se codifica como una acción, por ejemplo, la entrada de menú Salir (depende de Aplicación) debe ejecutar la acción de cerrar la aplicación, por lo que la conectaremos al método close de la siguiente manera:**
2. **action\_Salir = new QAction(MainWindow);**
3. **action\_Salir.setObjectName("action\_Salir");**
4. **action\_Salir.triggered.connect(MainWindow, "close()");**

**Y para la acción Reservas, que abre el dialogo de reservas del hotel tendremos que llamar al método abrir() que hemos creado antes de la siguiente manera:**

**action\_Reservas = new QAction(MainWindow);**

**action\_Reservas.setObjectName("action\_Reservas");**

**action\_Reservas.triggered.connect(this, "abrir()");**